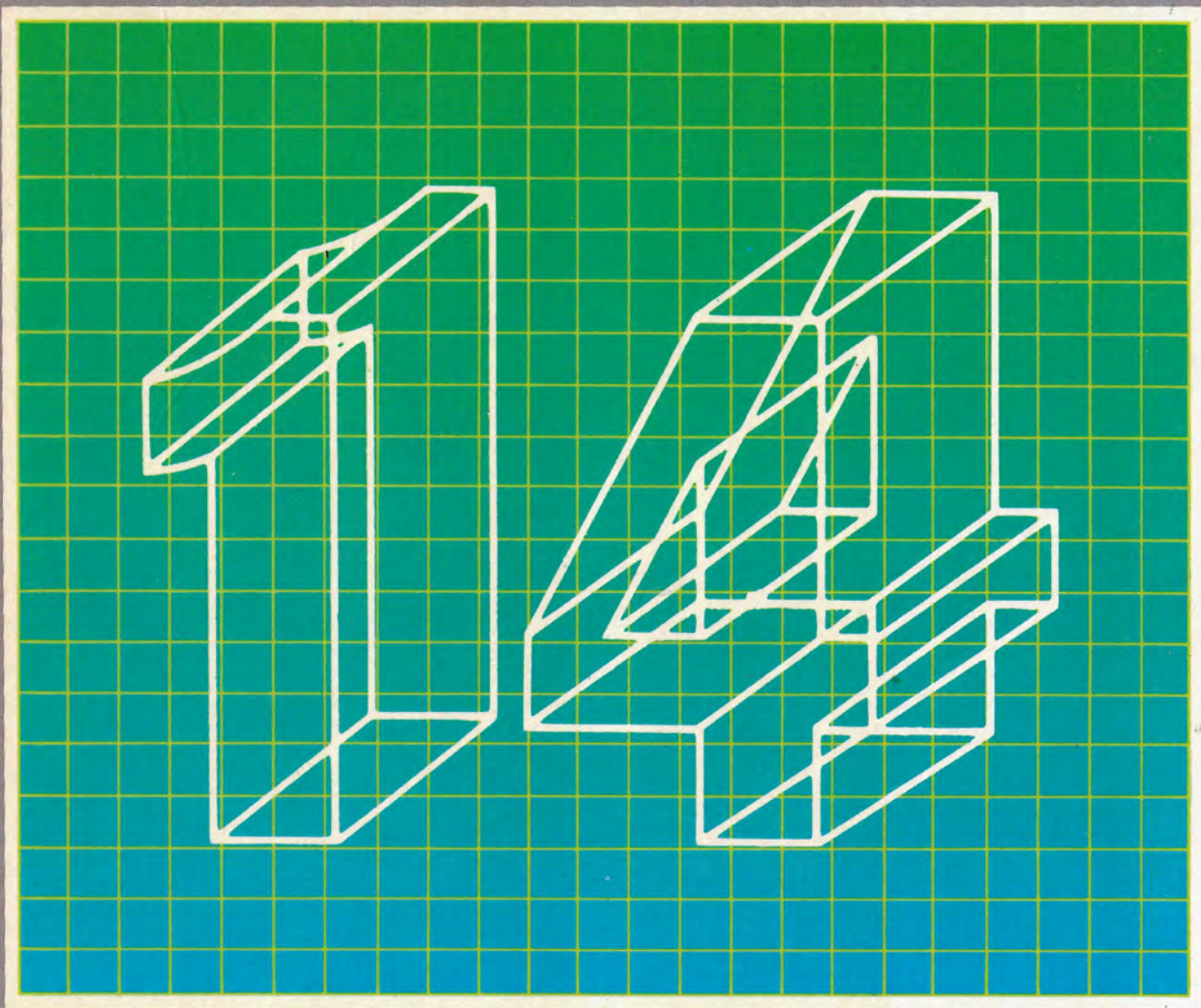


BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



LOS TEMAS MAS INTERESANTES

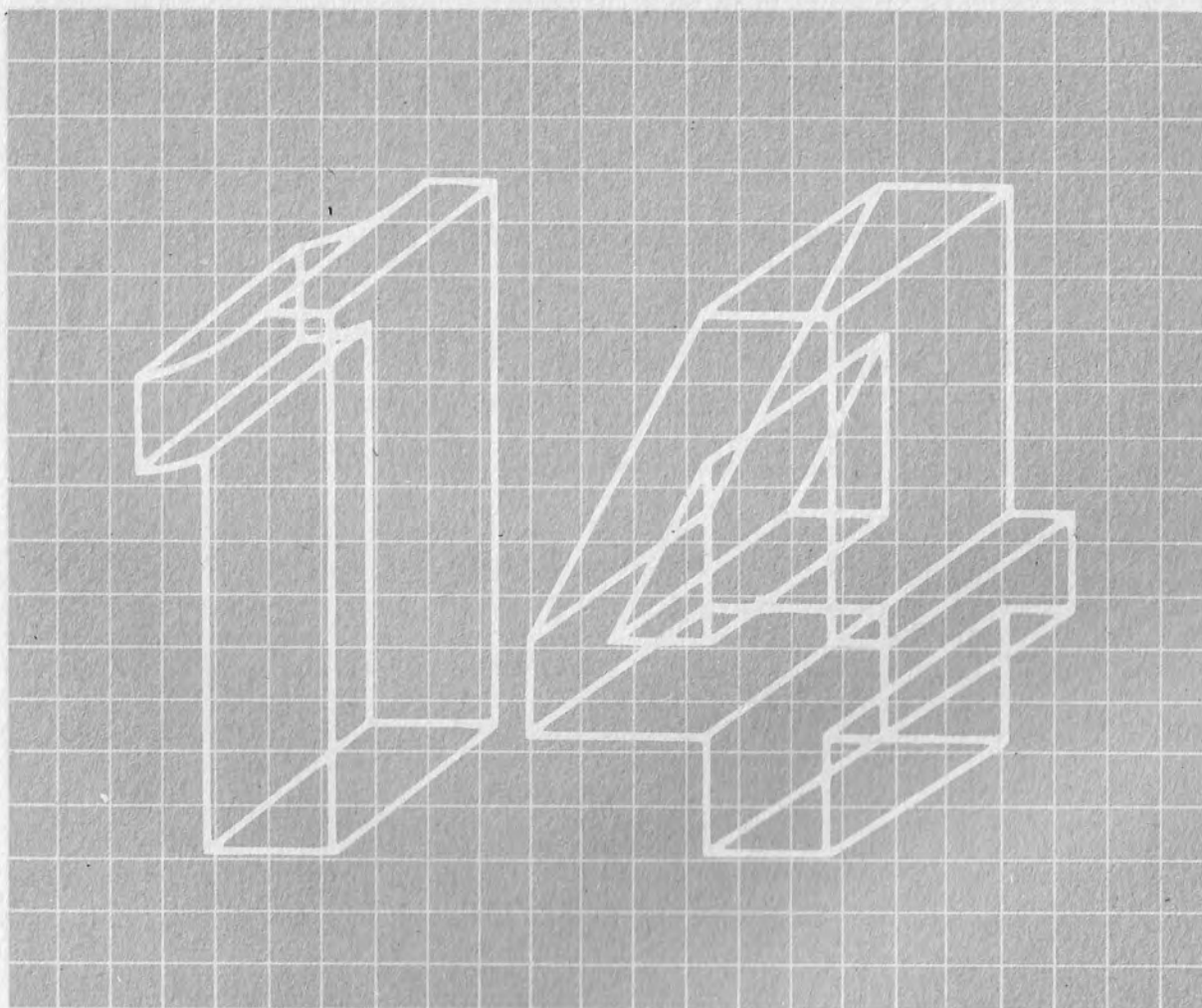
LOS PROGRAMAS MAS CURIOSOS

EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:

RICARDO ESPAÑOL CRESPO.

Gerente:

ANTONIO G. CUERPO.

Directora de producción:

MARIA LUISA SUAREZ PEREZ.

Director de la colección:

JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:

BRAVO-LOFISH.

Maquetación:

D. S. M.

Dibujos:

JOSE OCHOA

Tomo 14. «Experiencias y prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMÁTICA APLICADA: Soledad Tamariz-Martel, Diplomada en Telecomunicación; Francisco Blanca, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Fernando Suero, Diplomado en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Publicidad:

Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:

COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.

Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:

CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.

Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-086-3

ISBN de la obra: 84-7688-047-2

Fotocomposición:

ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:

MATEU CROMO. PINTO (Madrid).

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-43.185-1986

Printed in Spain - Impreso en España.

Suscripciones y números atrasados:

Ediciones Siglo Cultural, S. A.

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Mayo, 1987

P.V.P. Canarias: 355.—Ptas.

INDICE

■ EXPERIENCIA Y PRACTICAS EN LOGO	5
■ MANEJO DE SPRITES Y ELEMENTOS GRAFICOS	15
■ TRUCOS Y RUTINAS BASICAS	25
■ EL TALLER DEL HARDWARE	31
■ APRENDER CON EL ORDENADOR	43
■ PEQUEÑA HISTORIA DE LA INFORMATICA	51
■ TEMAS MONOGRAFICOS DE VANGUARDIA	55
■ TERMINOLOGIA	58
■ VOCABULARIO DE INFORMATICA	60

EXPERIENCIA Y PRACTICAS EN LOGO

■ Algo más sobre el manejo de procedimientos dentro del editor



AMOS a ver qué es lo que ocurre si actuamos de diferentes formas sobre los procedimientos cuando los tenemos en el editor del Logo. Una vez que tenemos editado un procedimiento, lo borramos por completo utilizando la tecla DELETE o BS. Borramos la orden PARA, el nombre del procedimiento, las órdenes que lo componen y el FIN. Si a continuación salimos del editor, ¿se habrá borrado el procedimiento de la memoria habiéndolo perdido para siempre? No, esto no ha ocurrido.

Si nos fijamos, en esta ocasión al salir del editor no hemos recibido el mensaje de "Nombre Procedimiento DEFINIDO", lo que significa que no ha habido actuación alguna sobre el procedimiento borrado, por lo que solamente habrá desaparecido del editor y no de la memoria.

Si borramos todo, a excepción de la orden PARA y del nombre que recibe el procedimiento, y salimos del editor, si lo habremos perdido para siempre, ya que ahora el procedimiento se vuelve a definir, pero en esta ocasión en blanco, sin contener orden alguna.

Ahora cambiamos el nombre de un procedimiento, que tenemos dentro del editor,

por otro diferente. Cuando salimos del editor, recibimos el mensaje que el procedimiento, que recibe el nuevo nombre que le hemos dado, queda definido. Pero, y el procedimiento anterior ¿se habrá borrado? No, simplemente se ha definido uno nuevo, copia exacta del anterior, que recibe un nombre diferente.

Supongamos ahora que tenemos un procedimiento definido llamado EJEMPLO. Si lo editamos de la siguiente forma:

```
? ED " EJEMPLO
```

Entraremos en el editor y en él aparecerá:

```
PARA
FIN
```

Y el procedimiento EJEMPLO, ¿se habrá borrado? ¿Por qué no aparece en el editor?

No se habrá borrado, lo que ha ocurrido es que a la hora de pedir la edición hemos dejado un espacio en blanco entre las comillas y el nombre del procedimiento, el cual se ignora, por lo que se interpreta que se solicita la entrada al editor sin la edición de un procedimiento.

Esto mismo ocurría cuando realizábamos lo siguiente:

```
? ED "
```

Ya hemos visto algunas de las posibilidades que nos ofrece el editor del Logo en cuanto al manejo de procedimientos. El conocimiento del buen funcionamiento de una de las

Internamente, el ordenador trabaja en código binario.

EXPERIENCIA Y PRACTICAS EN LOGO

herramientas más importantes que posee el Logo nos permite trabajar con seguridad y rapidez.

■ Copiar procedimientos

Hemos visto anteriormente cómo podemos obtener un procedimiento que sea copia de otro con sólo cambiarle el nombre cuando nos encontramos dentro del editor.

El Logo nos permite realizar este mismo proceso con la utilización de la orden:

COPIADEF

abreviatura de COPIA DEFINición.

Para utilizar esta orden hay que especificar dos nombres. El del procedimiento que se desea copiar y el que queremos que reciba la copia:

? COPIADEF "Nombre Procedimiento 1
"Nombre Procedimiento 2

Dependiendo del ordenador, esta operación se realiza de diferentes formas:

MSX y PC-COMPATIBLES

? COPIADEF "Nombre Procedimiento 1
"Nombre Procedimiento 2

El Nombre Procedimiento 1 corresponde al procedimiento que existe y que se quiere copiar y el Nombre Procedimiento 2, el nombre que va a recibir la copia.

SPECTRUM

? COPIADEF "Nombre Procedimiento 1
"Nombre Procedimiento 2

En este caso la operación se realiza al contrario. El Nombre Procedimiento 1 es el nombre que va a recibir el nuevo procedimiento y el Nombre Procedimiento 2, el que se va a copiar.

En ambos casos, el procedimiento que se copia no se borra.

Con los siguientes procedimientos, conseguimos señalar en un reloj, que previamente dibujamos, todas las horas del día.

Primero, definimos el procedimiento que dibuja el reloj sobre una peana. En la esfera aparecen las rayas indicando la división de la hora de cinco en cinco minutos:

? PARA RELOJ
> SL

```
> PONY -60
> GI 90
> BL
> AV 40 GI 90 AV 10
> GD 90 AV 15 GD 90
> AV 20 GD 90 AV 110
> GD 90 AV 20 GD 90
> AV 15 GD 90 AV 10
> GI 90 AV 40
> SL
> PONPOS [-55 -50]
> BL
> PONRUMBO 45
> AV 39 GD 45
> SL
> AV 76
> BL
> AV 34 GD 135 AV 39
> GI 45 AV 20 GI 135
> AV 39 GI 45 AV 20
> SL
> PONPOS [35 0]
> PONRUMBO 0
> BL
> REPITE 36 [AV 8 GD 10]
> SL
> PONX -29
> BL
> REPITE 36 [AV 5 GD 10]
> SL
> PONX
> BL
> RELLENA
> SL
> PONPOS [0 3]
> HAZ "A 0
> REPITE 12 [PONRUMBO :A SL AV 20
BL AV 3 SL
  PONPOS [0 3] HAZ "A + 30]
> FIN
```

A continuación definimos el procedimiento AGUJAS con el que vamos a dibujar la aguja de las horas y de los minutos, indicando la hora que deseemos:

```
? PARA AGUJA :HORAS :MINUTOS
> PONRUMBO :HORAS * 30 + (:MINUTOS / 2)
> BL
> AV 15
> GI 150
> REPITE 3 [AV 4 GI 120]
```

El código ASCII relaciona a todos los caracteres que gestiona un ordenador con un número entero.

```

> SL
> PONPOS [0 3]
> BL
> PONRUMBO :MINUTOS * 6
> AV 18
> GI 150
> REPITE 3 [AV 4 GI 120]
> FIN

```

Por último, definimos el procedimiento HORA, con el que inicializamos el proceso y en donde se hace una llamada a los otros dos procedimientos:

Este pasa al procedimiento AGUJAS la hora que nosotros le indiquemos:

```

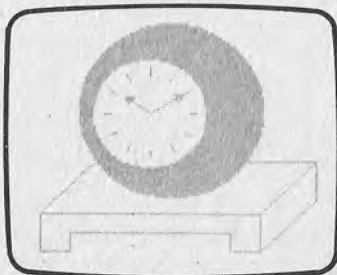
? PARA HORA :HORAS :MINUTOS
> BP
> OT
> RELOJ
> AGUJA :HORAS :MINUTOS
> FIN

```

A la hora de ejecutarlo, tenemos que indicar, en primer lugar, la hora y a continuación los minutos. Las horas después de las doce de la mañana se pueden indicar con dos cifras, por ejemplo, las ocho de la tarde se puede expresar como las 20 horas.

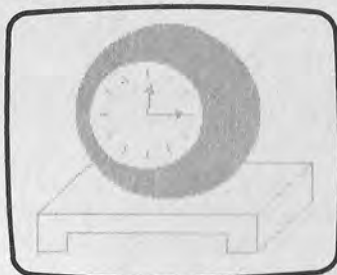
Para obtener el reloj marcando las 10 y 10:

```
? HORA 10 10
```



Para que marque las 12 y cuarto:

```
? HORA 12 15
```



■ Tratamiento de caracteres

Cuando asignamos un valor a una variable, estamos introduciendo un objeto en una parte de la memoria, que se identifica con el nombre que le damos a la variable.

Este objeto para poder ser almacenado en memoria debe pasar primero por un proceso de codificación, pues el ordenador internamente gestiona los datos en forma de ceros o unos, es decir, en código binario.

Si el objeto es un valor numérico se codificará, antes de almacenarse, en base dos (ceros y unos).

Si el objeto es uno o varios caracteres, éstos se codificarán en un código especial para datos alfanuméricos: el código ASCII.

El código ASCII relaciona todos los caracteres (especiales, numéricos y letras) con un número decimal.

Hay momentos en el desarrollo de un procedimiento o de un programa en los que nos interesa trabajar con los códigos ASCII de los caracteres, el LOGO ya lo tiene previsto, por ello dispone de las primitivas ASCII y CAR.

— La primitiva ASCII devuelve el valor del código ASCII del carácter que indiquemos. En el caso de una cadena de caracteres devuelve el valor del código del primer carácter.

Aunque en la tabla de códigos ASCII, que mostramos más adelante, sólo hayamos especificado 128 códigos, algunos ordenadores disponen de hasta 255 códigos. Por tanto, dependiendo del ordenador la primitiva ASCII devolverá un entero comprendido entre 0 y 127, o entre 0 y 255.

```
? ESCRIBE ASCII "A
65
```

```
? ESCRIBE ASCII "H
72
```

```
? ESCRIBE ASCII "CASA
67
```

el entero 67 corresponde al código ASCII de la primera letra de la cadena (C).

```
? ESCRIBE ASCII "0
48
```

```
? ESCRIBE ASCII "4
52
```

El Logo posee primitivas para gestionar la tabla de códigos ASCII.

EXPERIENCIA Y PRACTICAS EN LOGO

```
? ESCRIBE ASCII "d
100
```

```
? HAZ "A "HOLA
? ESCRIBE ASCII :A
72
```

```
? HAZ "A "12
? ESCRIBE ASCII :A
49
```

```
? HAZ "A ASCII "E
? ESCRIBE :A
69
```

```
? HAZ "A (ASCII "A) + (ASCII "B)
? ESCRIBE :A
131
```

Para obtener los códigos ASCII de las letras mayúsculas, definimos este procedimiento:

```
? PARA "MAYUSCULAS
> HAZ "I 0
> HAZ "A ASCII "A
> REPITE 26 [ESCRIBE :A HAZ "I :A + 1
HAZ "A :I]
> FIN
```

Los códigos de las letras minúsculas con el procedimiento MINUSCULAS:

```
? PARA MINUSCULAS
> HAZ "I 0
> HAZ "A ASCII "a
> REPITE 26 [ESCRIBE :A HAZ "I :A + 1
HAZ "A :I]
> FIN
```

También podemos definir un procedimiento de dos entradas, una para determinar la primera letra de la que queremos obtener el código y otra para la última letra:

```
? PARA GEN :A :B
> HAZ "I 0
> HAZ "A ASCII :A
> HAZ "B ASCII :B
> HAZ "R :B - :A + 1
> REPITE :R [ESCRIBE :A HAZ "I :A + 1
HAZ "A :I]
> FIN
```

Lo ejecutamos, por ejemplo:

```
? GEN "A "Z
```

y obtendremos los códigos ASCII de las letras de la A a la Z, ambas inclusive.

La primitiva CAR devuelve el carácter que le corresponde al código ASCII que indiquemos.

```
? ESCRIBE CAR 65
A
```

```
? ESCRIBE CAR "A
CAR NO ACEPTA A COMO
DATO
```

el dato debe ser un entero, comprendido entre 0 y 127, ó 0 y 255, según el ordenador con el que trabajemos.

```
? ESCRIBE CAR 300
CAR NO ACEPTA 300 COMO
DATO
```

```
? ESCRIBE CAR —2
CAR NO ACEPTA —2 COMO
DATO
```

```
? ESCRIBE CAR 49
1
```

```
? ESCRIBE CAR 125
}
```

```
? ESCRIBE CAR (SUMA 65 1)
B
```

```
? ESCRIBE CAR 65 + 2
C
```

```
? ESCRIBE CAR 65 + 3
D
```

```
?HAZ "A 50
? ESCRIBE CAR :A
2
```

Te aconsejamos que cuando te encuentres con expresiones complicadas utilices los paréntesis, aunque sólo sea para clarificar.

Podemos realizar con CAR el proceso inverso al que hemos realizado con ASCII, obtener los caracteres que correspondan a los enteros que indiquemos.

Los 32 primeros códigos son caracteres de control. No todos ellos son accesibles desde el LOGO.

Junto al carácter que les corresponde a los 32 primeros códigos hemos indicado una combinación de teclas con las que podemos acceder a ellos.

Para clarificar expresiones utilizamos los paréntesis.

Tabla de códigos ASCII

C. decimal	Carácter	C. decimal	Carácter	C. decimal	Carácter	C. decimal	Carácter
0	^	NUL	32	espacio	64		96
1	^A	SOH	33	!	65	A	97
2	^B	STX	34	"	66	B	98
3	^C	ETX	35		67	C	99
4	^D	EOT	36	ñ	68	D	100
5	^E	ENQ	37	%	69	E	101
6	^F	ACK	38	&	70	F	102
7	^G	BEL	39	'	71	G	103
8	^H	BS	40	(	72	H	104
9	^I	HT	41	)	73	I	105
10	^J	LF	42	*	74	J	106
11	^K	VT	43	+	75	K	107
12	^L	FF	44	,	76	L	108
13	^M	CR	45	—	77	M	109
14	^N	SO	46	.	78	N	110
15	^O	SI	47	/	79	O	111
16	^P	DLE	48	0	80	P	112
17	^Q	DC1	49	1	81	Q	113
18	^R	DC2	50	2	82	R	114
19	^S	DC3	51	3	83	S	115
20	^T	DC4	52	4	84	T	116
21	^U	NAK	53	5	85	U	117
22	^V	SYN	54	6	86	V	118
23	^W	ETB	55	7	87	W	119
24	^X	CAN	56	8	88	X	120
25	^Y	EM	57	9	89	Y	121
26	^Z	SUB	58	:	90	Z	122
27	^¿	ESC	59	:	91	¿	123
28	^/	FS	60	<	92	/	124
29	^]	GS	61	=	93	] ^	125
30	^~	RS	62	>	94	~	126
31	^_	US	63	?	95	_	127
							DEL

El símbolo ^ significa CTRL, por tanto, ^A lo conseguimos pulsando simultáneamente las teclas CTRL y A.

■ Cuadro resumen

— ASCII "carácter"

Esta primitiva devuelve el valor del código ASCII del carácter que indiquemos; en el caso de varios caracteres, devuelve el valor del código ASCII del primero.

Dependiendo del ordenador, esta primitiva devolverá un número comprendido entre 0 y 127, ó 0 y 255.

— CAR valor del código ASCII.

La primitiva CAR devuelve el carácter que corresponde al código ASCII que indiquemos.

El número que indiquemos deber ser un entero comprendido entre 0 y 127, ó 0 y 255 (dependerá del ordenador).

— COPIADEF "Nombre Proc. antiguo"

"Nombre Proc. nuevo."

Hace una copia del procedimiento que

Puedo realizar copias de los procedimientos.

EXPERIENCIA Y PRACTICAS EN LOGO

se especifica en primer lugar en otro procedimiento que va a recibir el nombre que se especifica en segundo lugar.

Ejercicios

1. *Dibuja una casa y coloréala. Utiliza para dibujar y colorear la casa un procedimiento de tantas entradas como partes a colorear tenga la casa.*
2. *Define un procedimiento que escriba a través del código ASCII las letras mayúsculas, otro para las letras minúsculas, y, por último, uno general para obtener los caracteres que correspondan a un intervalo de códigos, que se especificarán con variables de entrada.*
3. *Define los procedimientos necesarios para poder dibujar los números romanos del 1 al 10.*
4. *Define cinco procedimientos para que la Tortuga cuando en vez de ir dejando una simple línea tras de sí, deje los siguientes tipos de líneas:*
 - : línea doble.
 - : línea de puntos.
 - : línea de rayas de dos puntos de longitud y separación de uno.
 - : línea de punto y raya.
 - : línea de rayas paralelas de dos puntos de longitud y separadas un punto.
5. *¿Son correctas las órdenes siguientes?:*
 - ? ESCRIBE CAR ASCII "A
 - ? HAZ "A 20
 - ? HAZ "B 60
 - ? ESCRIBE CAR :A + :B
 - ? HAZ "A ASCII "A + ASCII "B
 - ? COPIADEF "CASA CASONA
 - ? ED ":V
 - ? PARA CASA :V:A

Solución a los ejercicios

1:

Nosotros hemos dibujado una casa en la que coloreamos cuatro partes, las ventanas (del mismo color), la puerta, el tejado y la fachada.

:T color del tejado
:C color de la fachada
:P color de la puerta
:V color de las ventanas

```
? PARA CASA :T :C :P :V
> BP
> CENTRO
> BL
> REPITE 2 [AV 40 GD 90 AV 38 GD 90]
> AV 40 GD 50 AV 25
> GD 80 AV 25
> SL
> PONX 15 PONY 0
> BL
> PONRUMBO 0 AV 10
> REPITE 18 [AV 1 GD 10]
> AV 10
> HAZ "PX COORX
> HAZ "PY COORY
> SL
> PONX (:PX — 3)
> PONY (:PY + 3)
> BL
> PONCL :P
> RELLENA
> SL
> PONX 5
> PONY 27
> BL
> PONRUMBO 90
> VENTA :V
> SL
> AV 21
> BL
> VENTA :V
> SL
> PONPOS [1 1]
> BL
> PONCL :C
> RELLENA
> SL
> PONPOS [10 42]
> BL
> PONCL :T
> RELLENA
> FIN
```

Dentro del procedimiento CASA hacemos dos llamadas al procedimiento VENTA para dibujar las ventanas, indicando el color (V) con el que vamos a pintarlas.

```
? PARA VENTA :V
```

Haz copias de seguridad de los procedimientos.

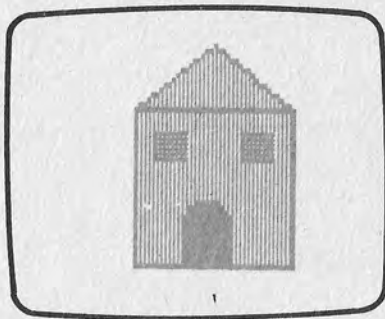
```

> PONCL :V
> REPITE 2 [AV 7 GI 90 AV 7 GI 90]
> SL
> PONX COORX + 2
> PONY COORY + 2
> BL PONCL :V
> RELLENA
> SL
> PONX COORX — 2
> PONY COORY — 2
> FIN

```

Dibujamos y pintamos la casa:

? CASA 3 3 1 2



cambiando los colores

? CASA 1 2 3 1

2:

Con MAYUSCULAS obtenemos las 26 letras mayúsculas, indicando los códigos ASCII con una variable que vamos incrementando en uno.

```

? PARA MAYUSCULAS
> HAZ "A 65
> HAZ "I 0
> REPITE 26 [ESCRIBE CAR :A HAZ "I :A + 1
    HAZ "A :I]
> FIN

```

Con MINUSCULAS obtenemos todas las letras minúsculas.

```

? PARA MINUSCULAS
> HAZ "A 97
> HAZ "I 0
> REPITE 26 [ESCRIBE CAR :A HAZ "I :A + 1
    HAZ "A :I]
> FIN

```

Y con el procedimiento de dos entradas GENERAL obtenemos los caracteres que corresponden a un intervalo de códigos que especificamos cuando ejecutamos el procedimiento.

:A para indicar el primero de los códigos
:B para indicar el último de los códigos

? PARA GENERAL :A :B

```

> HAZ "C0
> HAZ "R :B — :A + 1
> REPITE :R [ESCRIBE CAR :A HAZ "C :A +
    1 HAZ "A :C]
> FIN

```

Una vez definido, si queremos saber qué caracteres corresponden a los códigos comprendidos entre el 50 y el 60:

? GENERAL 50 60

3:

Para representar los números romanos del 1 al 10 sólo es necesario utilizar los siguientes signos:

- I: Que representa el 1
- V: Que representa el 5
- X: Que representa el 10

Los demás números se obtienen por combinación de estos signos.

Definimos tres procedimientos que dibujen la I, la V y la X:

? PARA I

```

> BL
> PONRUMBO —90
> REPITE 2 [ REPITE 3 [AV 6 GD 90] GI 180
    AV 18 GI 90 REPITE 2 [AV 6 GD 90] AV 12]
> SL
> PONX COORX + 25
> PONY 0
> PONRUMBO 0
> FIN

```

? PARA V

```

> BL
> PONRUMBO —90
> AV 3 DG 80 AV 30
> PONRUMBO 90
> AV 5 GD 80 AV 25
> GI 160 AV 25
> PONRUMBO 90

```

Si borramos por completo un procedimiento que esté en el editor, éste no se borra de la memoria.

EXPERIENCIA Y PRACTICAS EN LOGO

```
> AV 5 GD 103 AV 30
> PONRUMBO '—90
> AV 4
> SL
> PONX COORX + 25
> PONY 0
> PONRUMBO 0
> FIN
```

? PARA X

```
> BL
> REPITE 2 [ GD 25 AV 33 GD 65 AV 5 GD 90]
> SL
> GD 90 AV 14 GI 90
> BL
> REPITE 2 [GI 25 AV 33 GD 115 AV 5 GD 90]
> SL
> PONX COORX + 25
> PONY 0
> PONRUMBO 0
> FIN
```

Los tres signos tienen igual altura y en todos ellos la Tortuga queda situada para dibujar otro signo a una cierta distancia.

Ahora definimos los procedimientos para dibujar los números romanos:

? PARA UNO

```
> BP
> I
> FIN
```

? PARA DOS

```
> BP
> I
> I
> FIN
```

? PARA TRES

```
> BP
> I
> I
> I
> FIN
```

? PARA CUATRO

```
> BP
> I
> V
> FIN
```

? PARA CINCO

```
> BP
> V
> FIN
```

? PARA SEIS

```
> BP
> V
> I
> FIN
```

? PARA SIETE

```
> BP
> V
> I
> I
> FIN
```

? PARA OCHO

```
> BP
> V
> I
> I
> I
> FIN
```

? PARA NUEVE

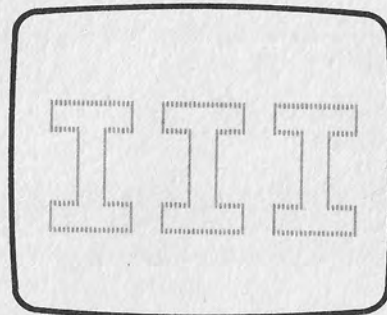
```
> BP
> I
> X
> FIN
```

? PARA DIEZ

```
> BP
> X
> FIN
```

Si queremos representar el número romano tres:

? TRES



Utiliza nombres de variables cortos.

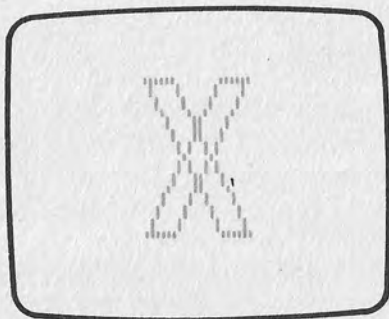
El ocho:

? OCHO



El diez:

? DIEZ



4:

Definimos los cinco procedimientos en los que se utilizan como variable de entrada :LONG, que es la que determina lo que tiene que avanzar la Tortuga:

Para la línea de doble ancho:

```
? PARA AVA :LONG
> BL
> AV :LONG
> GD 90 AV 1 GI 90
> RE :LONG — 1
> AV :LONG — 1
> FIN
```

Para la línea de puntos:

```
? PARA AVB :LONG
> BL
> REPITE :LONG / 4 [BL AV 2 SL AV 2]
> FIN
```

Para la línea de rayas:

```
? PARA AVC :LONG
> BL
> REPITE :LONG / 3 [BL AV 2 SL AV 1]
> FIN
```

Para la línea de puntos y rayas:

```
? PARA AVD :LONG
> BL
> REPITE :LONG / 5 [AV 2 SL AV 1 BL
AV 1 SL AV 1 BL]
> FIN
```

Para la línea de rayas paralelas:

```
? PARA AVE :LONG
> BL
> REPITE :LONG / 2 [BL GD 90 AV 2 RE
2 GI 90 SL AV 2]
> FIN
```

Una vez definidos todos los procedimientos, definimos uno general para que se muestre en la pantalla todos los tipos de línea:

```
? PARA MODELOS :LONG
> BP
> OT
> SL
> PONPOS [—20 0]
> BL
> AVA :LONG
> SL
> PONPOS [—10 0]
> BL
> PONRUMBO 0
> AVB :LONG
> SL
> PONPOS [0 0]
> BL
> PONRUMBO 0
> AVC :LONG
> SL
> PONPOS [10 0]
> BL
> PONRUMBO 0
> AVD :LONG
> SL
> PONPOS [20 0]
> BL
> PONRUMBO 0
> AVE :LONG
> FIN
```

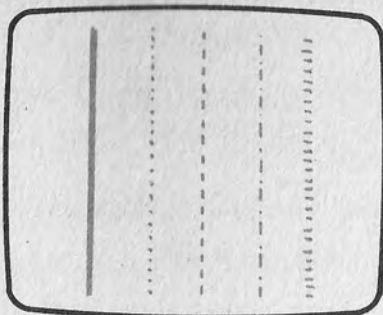
Puedes utilizar el mismo nombre de variables en distintos procedimientos.

EXPERIENCIA Y PRACTICAS EN LOGO

Si queremos ejecutarlo, basta con dar un valor a la variable :LONG, el cual pasará a todos los procedimientos.

? MODELOS 50

Y en la pantalla obtendremos:



5:

— ? ESCRIBE CAR (ASCII "A)
CORRECTO

Obtenemos el mismo carácter que hemos indicado en ASCII la A.

— ? HAZ "A 20
? HAZ "B 60
? ESCRIBE CAR :A + :B
CORRECTO

Obtenemos el carácter que le corresponde al código 80 (20 + 60), que es la letra P.

— ? HAZ "A ASCII "A + ASCII "B
INCORRECTO.

El LOGO muestra el mensaje:

+ NO ACEPTA A COMO DATO

En expresiones de este tipo te aconsejamos que utilices los paréntesis.

? HAZ "A (ASCII "B) + (ASCII "C)
? ESCRIBE :A
133

— ? COPIADEF "CASA CASONA
INCORRECTA.

Falta indicar las comillas en el segundo nombre de procedimiento

— ? ED ":V
CORRECTO.

En este caso :V está actuando como nombre de procedimiento, y no de variable.

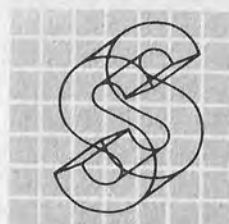
— PARA CASA :V:A
CORRECTO.

Al no dejar espacio se define una sola variable con el nombre :V:A.

Una variable global tiene que tener un valor inicial para poder utilizarla.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

■ La animación

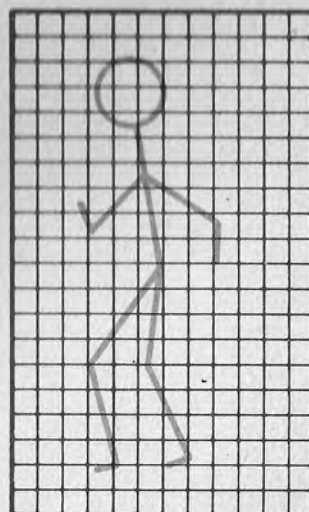
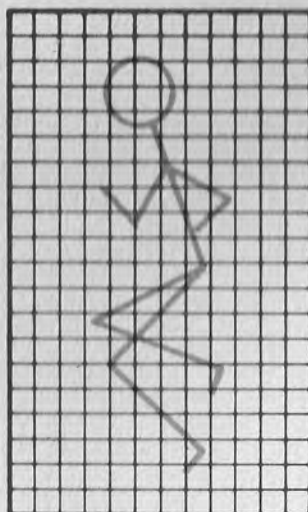
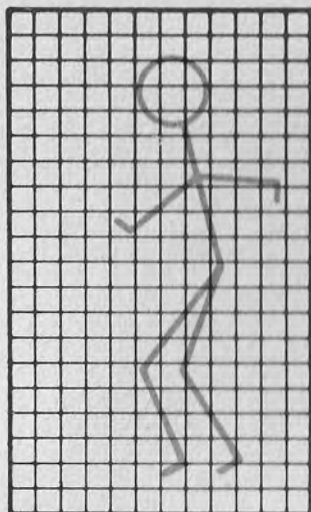
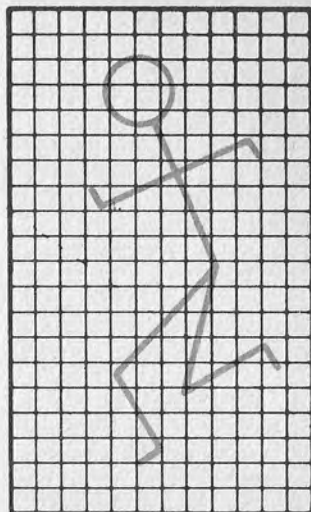


Se puede decir que todos los programas de juegos tienen algo de animación. En algún momento del programa aparece un personaje o un objeto que se mueve por la pantalla. Esto no suele ocurrir en los llamados juegos de mesa, como puede ser el tute o el dominó, ya que no necesitan que los gráficos sean animados. Cualquier otro juego que nos inventemos tendrá una parte de animación gráfica.

Podemos decir que la parte más importante de un programa es la animación gráfica. Es la más importante y también la más difícil de realizar. Siempre que deseemos realizar un programa que tenga animación y movimiento, tenemos que cuidar lo más posible que ésta

sea del agrado del usuario, ya que es lo primero en lo que un usuario se fija.

A la hora de mover un personaje o un objeto por la pantalla tenemos dos maneras de hacerlo. Una de ellas es, como hemos visto hasta ahora, dibujando el objeto o personaje en una posición de la pantalla, borrarlo, volverlo a imprimir en una posición distinta de la anterior, volverlo a borrar, etc. De esta manera, aunque el objeto se mueva por la pantalla, no cambia de forma con el transcurso del tiempo. Si la figura que estamos moviendo es un hombre, entonces no nos dará la impresión de que esté andando, nos parecerá que se desliza. La otra forma de mover un personaje por la pantalla es moverlo de forma que, si se trata de un hombre, nos dé la impresión de que está hablando. Para conseguir esto necesitamos tener varios dibujos del mismo hombre con posiciones distintas al andar. A la hora de realizar el movimiento, imprimiremos el pri-



MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

mero de los dibujos, lo borraremos, imprimiremos el segundo de los dibujos (que tendrá un pie un poco más levantado), lo borraremos y así sucesivamente hasta que se haya completado un movimiento entero.

Si nos fijamos en la figura 1 podemos ver las diferentes posturas de las que se puede componer el paso de un hombre al andar.

En este tomo y en los siguientes iremos viendo cómo realizar animación en nuestro ordenador.

Animación de una figura

Cuando vimos el movimiento lo dividimos en dos clases; movimiento de figuras sencillas (compuestas de un solo carácter) y movimiento de figuras complejas (compuestas por dos o más caracteres). Vamos a dividir la animación en dos partes. Primero veremos cómo mover una sola figura por la pantalla y más tarde cómo realizar programas que nos gestionen el movimiento de dos a más figuras

```
1000 REM *****
1010 REM * MOVIMIENTO ANIMADO DE UN *
1020 REM * COCHE CON CHOQUE AL LLE- *
1030 REM * GAR AL BORDE DE LA PANTA *
1040 REM * LLA. *
1050 REM *****
1060 REM
1070 REM *** DEFINICION DEL COCHE ***
1080 REM
1090 DIM A$(3,2)
1100 REM
1110 REM *** POSICION NUMERO UNO ***
1120 REM
1130 LET A$(1,1)="/====\"
1140 LET A$(1,2)=" O O "
1150 REM
1160 REM *** POSICION NUMERO DOS ***
1170 REM
1180 LET A$(2,1)="\\====/"
1190 LET A$(2,2)=" o o "
1200 REM
1210 REM *** POSICION DE CHOQUE ***
1220 REM
1230 LET A$(3,1)="/==\"
1240 LET A$(3,2)="Oo=="
1250 REM
1260 REM *** PROGRAMA GENERAL ***
1270 REM
1280 REM *** DIBUJO DE LA CARRETERA ***
1290 REM
1300 CLS
1310 LOCATE 10,1
1320 FOR I=1 TO 40
1330 PRINT "*";
1340 NEXT I
1350 REM
1360 REM *** DIBUJO DE LA PARED ***
1370 REM
1380 LOCATE 5,1
1390 FOR I=1 TO 5
1400 PRINT "*"
1410 NEXT I
1420 REM
1430 REM *** DIBUJO DE LA POSICION INICIAL DEL COCHE ***
1440 REM
1450 LOCATE 8,32
1460 PRINT A$(2,1)
1470 LOCATE 9,32
1480 PRINT A$(2,2)
1490 LOCATE 15,1
1500 RR=40
```

```

1510 PRINT "PULSA UNA TECLA PARA EMPEZAR"
1520 REM
1530 REM *** PULSA UNA TECLA Y HUMO DEL COCHE ***
1540 REM
1550 LET B$=INKEY$
1560 LOCATE 8,38
1570 PRINT ". "
1580 GOSUB 1910
1590 LOCATE 8,38
1600 PRINT ". "
1610 GOSUB 1910
1620 LOCATE 8,38
1630 PRINT ". "
1640 GOSUB 1910
1650 IF B$="" THEN GOTO 1550
1660 LOCATE 15,1
1670 PRINT " "
1680 REM " :REM 28 ESPACIOS
1690 REM *** MOVIMIENTO DEL COCHE ***
1700 REM
1710 LET RR=70
1720 FOR I=30 TO 2 STEP -2
1730   FOR J=2 TO 1 STEP -1
1740     LOCATE 8,I+J
1750     PRINT A$(J,1)
1760     LOCATE 9,I+J
1770     PRINT A$(J,2)
1780     GOSUB 1910
1790   NEXT J
1800 NEXT I
1810 REM
1820 REM *** CHOQUE ***
1830 REM
1840 LOCATE 8,2
1850 PRINT A$(3,1)
1860 LOCATE 9,2
1870 PRINT A$(3,2)
1880 LOCATE 12,1
1890 PRINT "AUCH !!!"
1900 END
1910 REM
1920 REM *** BUCLE DE RETARDO ***
1930 REM
1940 FOR K=1 TO RR
1950 NEXT K
1960 RETURN

```

moviéndose a la vez. Las figuras de las que vamos a hablar pueden ser sencillas o complejas, construidas con los caracteres definibles por el usuario o con los normales de la máquina.

Ya hemos visto algunos ejemplos de animación, aunque no los hemos comentado. El programa 1, que aparece a continuación, nos muestra un coche que se mueve por la pantalla y que se choca al llegar al borde.

Como siempre, y antes de explicar cómo funciona el programa, vamos a ver las variaciones que hay que hacer para que pueda funcionar en ordenadores distintos del IBM PC y compatibles:

COMMODORE:

```

1300 PRINT CHR$(147)
1310 POKE 214,9:POKE 211,0
1380 POKE 214,4:POKE 211,0
1450 POKE 214,7:POKE 211,31
1470 POKE 211,8:POKE 211,31
1490 POKE 214,14:POKE 211,0
1550 GET B$
1560 POKE 214,7:POKE 211,37
1590 POKE 214,7:POKE 211,37
1620 POKE 214,7:POKE 211,37
1660 POKE 214,14:POKE 211,0
1720 FOR I=29 TO 1 STEP -2
1740 POKE 214,7:POKE 211,I+J

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
1760 POKE 214,8:POKE 211,I+J
1840 POKE 214,7:POKE 211,1
1860 POKE 214,8:POKE 211,1
1880 POKE 214,11:POKE 211,0
```

SPECTRUM:

```
1090 DIM A$
1310 PRINT AT 9,0;
1320 FOR I=1 TO 32
1380 PRINT AT 4,0;
1450 PRINT AT 7,22;
1470 PRINT AT 8,22;
1490 PRINT AT 14,0;
1500 LET RR=40;
1560 PRINT AT 7,28;
1590 PRINT AT 7,28;
1620 PRINT AT 7,28;
1660 PRINT AT 14,0;
1720 FOR I=20 TO 1 STEP -2
1740 PRINT AT 7,I+J;
1760 PRINT AT 8,I+J;
1840 PRINT AT 7,1;
1860 PRINT AT 8,1;
1880 PRINT AT 11,0;
```

AMSTRAD y MSX:

```
1310 LOCATE 1,10
1380 LOCATE 1,5
1450 LOCATE 32,8
1470 LOCATE 32,9
1490 LOCATE 1,15
1560 LOCATE 38,8
1590 LOCATE 38,8
1620 LOCATE 38,8
1660 LOCATE 1,15
1740 LOCATE I+J,8
1760 LOCATE I+J,9
1840 LOCATE 2,8
1860 LOCATE 2,9
1880 LOCATE 1,12
```

Se recomienda a los usuarios del IBM y del AMSTRAD que ejecuten este programa con la pantalla en el modo de cuarenta caracteres por línea (WIDTH 40 en el IBM, MODE 2 en el AMSTRAD).



El funcionamiento del programa línea a línea, aun siendo muy sencillo gracias a la gran cantidad de líneas REM que tiene, es el siguiente:

Línea 1090. Definimos una matriz de 32 elementos. El coche que vamos a definir ocupa dos líneas y tiene tres posiciones distintas. Por ello, la matriz tiene tres filas con dos columnas cada fila. En el SPECTRUM esta línea es un poco diferente. El tercer número (el 7) le dice al ordenador la longitud máxima en caracteres que tiene cada elemento de la matriz.

Líneas 1130 y 1140. En estas dos líneas definimos la primera posición del movimiento del coche. En esta posición las ruedas son grandes como es la posición número 1, se almacena en la primera fila de la matriz. Cada línea del dibujo se almacena en una de las columnas de dicha primera línea.

Líneas 1180 y 1190. Definimos la segunda posición del coche. En este coche el caso tiene las ruedas más pequeñas aparte de otros cambios en su forma. Lo almacenamos en la segunda fila de la matriz.

Líneas 1230 y 1240. Almacenamos la tercera posición del coche, que es la posición de choque. En esta posición, el coche parece cualquier cosa menos un coche, pero al utilizar caracteres ya definidos en el ordenador, se hace más difícil encontrar una forma que sea realmente la que se busca. Esta última forma se almacena en la tercera fila de la matriz.

Línea 1300. Borrarnos la pantalla.

Línea 1310. Colocamos el cursor en la línea 10 de la pantalla. En esta línea dibujaremos el suelo por donde se va a mover el coche.

Línea 1320. Comenzamos un bucle entre 1 y 40 (entre 1 y 32 en el SPECTRUM) dentro del cual dibujaremos el suelo.

Línea 1330. Dibujamos el suelo a base de asteriscos.

Línea 1340. Aquí termina el bucle.

Línea 1380. Colocamos el cursor en la quinta línea de la pantalla para dibujar la pared con la que se tiene que chocar el coche.

Línea 1390. Al igual que un poco más arriba, comenzamos un bucle dentro del cual dibujaremos dicha pared.

Línea 1400. Y la dibujamos formada por asteriscos.

Línea 1410. Aquí termina el bucle.

Línea 1450. Colocamos el cursor al final de la octava línea de la pantalla para dibujar el coche parado.

Línea 1460. Imprimimos la primera fila

de las dos de que se compone el dibujo del coche.

Líneas 1470 y 1480. Repetimos lo que acabamos de hacer, pero con la siguiente línea del dibujo.

Línea 1490. Colocamos el cursor en la línea 15.

Línea 1500. Asignamos a la variable numérica RR el valor 40. Ya veremos más adelante cuál es la función de esta variable.

Línea 1510. Imprimimos un mensaje diciéndole al usuario que pulse una tecla.

Las líneas que vienen a continuación son las encargadas de mirar si el usuario ha pulsado alguna tecla. También tienen como función dibujar unos puntos en la pantalla, junto al coche, de manera que parezca que es el humo que está saliendo del tubo de escape.

Línea 1550. Miramos si se ha pulsado alguna tecla. En caso afirmativo, ésta queda almacenada en la variable B\$. La comprobación de si B\$ contiene algo se hace más abajo.

Línea 1560. Colocamos el cursor detrás del coche.

Línea 1570. E imprimimos un punto y seguido de dos espacios en blanco.

Línea 1580. Llamamos a la rutina que se encuentra a partir de la línea 1910 y cuya función es retener el programa durante un tiempo. En esta rutina se utiliza la variable RR. Esta nos dice el tiempo que se detendrá el programa.

Línea 1590. Volvemos a colocar el cursor en la posición anterior.

Línea 1600. E imprimimos un espacio en blanco, un punto y un carácter en blanco seguidos.

Línea 1610. Volvemos a llamar a la rutina de retardo.

Líneas 1620 a 1640. Repetimos la operación que acabamos de describir, pero esta vez imprimimos dos espacios en blanco seguidos de un punto.

Línea 1650. Si B\$ es igual a la cadena vacía (no se ha pulsado ninguna tecla), entonces volvemos a la línea 1550.

Gracias a este grupo de líneas podemos ver cómo va saliendo el humo del coche antes de que éste empiece a andar. Conseguimos dicho efecto dibujando un punto junto al coche y separándolo cada vez más.

El siguiente grupo de líneas son las que se encargan de producir el movimiento del

coche y de dibujarlo cuando se ha chocado contra la pared.

Línea 1710. Asignamos a la variable RR el valor 70. Esto es necesario porque esta parte del programa también hace uso de la rutina de retardo para que el coche no vaya muy de prisa. Variando el valor de esta variable podemos hacer que el coche vaya más o menos de prisa.

Línea 1720. Comenzamos un bucle que nos permitirá mover el coche de derecha a izquierda por la pantalla. Por eso, el incremento del bucle es negativo.

Línea 1730. Como el movimiento del coche está compuesto por dos dibujos, incluimos un nuevo bucle que va desde dos hasta uno, también con incremento negativo. En este caso, la elección de incremento negativo es arbitraria. Bien se podía haber hecho el bucle desde uno hasta dos, pero eso nos hubiese llevado a complicar las sentencias LOCATE (PRINT AT en el SPECTRUM) que aparecen a continuación.

Línea 1740. Colocamos el cursor donde corresponda. En cada vuelta del bucle se colocará más a la izquierda.

Línea 1750. Imprimimos la primera línea del dibujo que nos está indicando la variable J.

Línea 1760. Colocamos el cursor una línea más abajo.

Línea 1770. Y dibujamos la segunda línea del dibujo. En este punto ya tenemos impreso el coche en la pantalla.

Línea 1780. Llamamos a la rutina de retardo del tiempo.

Línea 1790. Y terminamos el segundo bucle. Al terminar el bucle, dibujaremos la otra posición del coche. Cuando este bucle termina. Cuando ha llegado a dos, el programa pasa a la siguiente línea para continuar con el siguiente valor del bucle principal y, por tanto, volver a comenzar este bucle. Con esto se consigue que vuelva a dibujarse de nuevo la primera posición del coche.

Línea 1800. Aquí termina el bucle principal.

Una vez que el bucle ha terminado, es que el coche está tocando a la pared. Este es el momento de imprimirlo como si se hubiese chocado.

Línea 1840. Colocamos el cursor en la segunda columna de la línea octava.

Línea 1850. E imprimimos la primera línea del dibujo del coche chocado.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

Líneas 1860 y 1870. Realizan la misma función que las dos líneas anteriores, pero con la segunda parte del coche chocado y en la siguiente línea.

Línea 1880. Colocamos el cursor unas cuantas líneas más abajo.

Línea 1890. E imprimimos un mensaje de dolor.

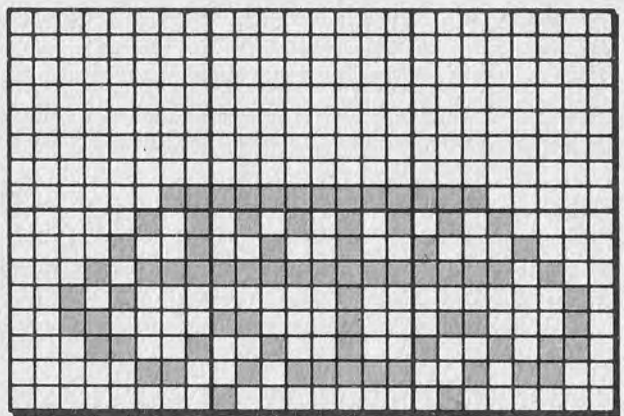
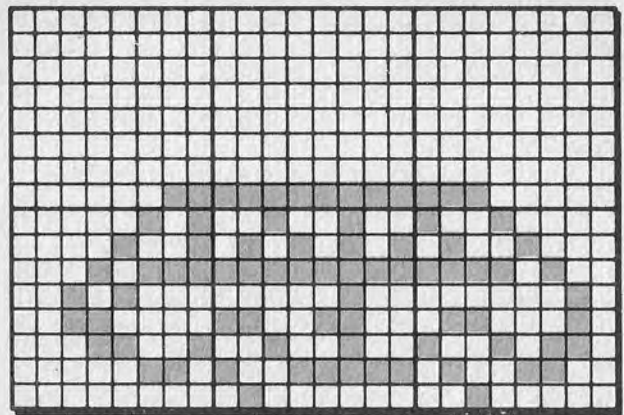
Entre las **líneas 1940 y 1960** se realiza el bucle de retardo del que ya hemos hablado. Hay que fijarse en que la variable tope del bucle es RR, y que dependiendo de que ésta tenga un valor más o menos grande, el bucle estará dando vueltas sobre sí mismo durante más o menos tiempo.

Este programa puede ilustrar perfectamente cómo hay que realizar el movimiento con animación en nuestro ordenador. La única pega que tiene el programa es que el coche no está muy conseguido y que, por tanto, no nos da mucha sensación de realidad. Por otra parte, el movimiento que hemos utilizado sólo se compone de dos posiciones distintas, lo cual le resta vistosidad, pues cuanto más posiciones utilizemos, más cerca de los dibujos animados nos encontraremos.

Se recomienda al lector que utilizando las capacidades de su ordenador para definir SPRITES o caracteres gráficos, varíe este programa de manera que el coche parezca un coche de verdad. A continuación, se dan los dibujos de cómo podrían ser los caracteres que habría que generar.

El programa también tiene otro fallo. Este consiste en que cuando el coche se está moviendo no despidе humo. El lector puede intentar variar el programa para que este fallo se resuelva antes de mirar la solución que se da a continuación.

•• Lo único que habría que cambiar para



que el coche echase humo sería cambiar las siguientes líneas:

```
1130 LET A$(1,1)="/====\"
1180 LET A$(2,1)="\====/"
1230 LET A$(3,1)="/==\"
```

Otra variación que se nos podía ocurrir sería que, después del choque el coche echase humo. Para realizar esto podemos utilizar la misma técnica que ya hemos utilizado en el programa. La única diferencia está en que el humo ha de ir hacia arriba. Para ello sólo tenemos que añadir las siguientes líneas:

```
1895 LET RR=40
1896 LOCATE 5,2
1897 PRINT " "
1898 LOCATE 6,2
1899 PRINT " "
1900 LOCATE 7,2
1901 PRINT " "
1902 GOSUB 1910
1903 LOCATE 5,2
1904 PRINT " "
1905 LOCATE 6,2
1906 PRINT " "
1907 LOCATE 7,2
1908 PRINT " "
1909 GOSUB 1910:GOTO 1896
```

Las variaciones de estas últimas líneas para los ordenadores que no sean IBM son las siguientes:

COMODORE:

```
1896 POKE 214,4:POKE 211,1
1898 POKE 214,5:POKE 211,1
1900 POKE 214,6:POKE 211,1
1903 POKE 214,4:POKE 211,1
1905 POKE 214,5:POKE 211,1
1907 POKE 214,6:POKE 211,1
```

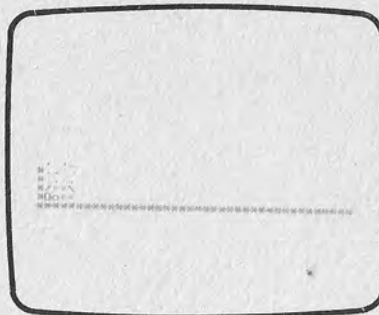
SPECTRUM:

```
1896 PRINT AT 4,1;
1898 PRINT AT 5,1;
1900 PRINT AT 6,1;
1903 PRINT AT 4,1;
1905 PRINT AT 5,1;
1907 PRINT AT 6,1;
```

AMSTRAD y MSX:

```
1896 LOCATE 2,5
1898 LOCATE 2,6
```

```
1900 LOCATE 2,7
1903 LOCATE 2,5
1905 LOCATE 2,6
1907 LOCATE 2,7
```



A continuación, aparece un programa que nos dibuja una cara en la pantalla. Podemos hacer que dicha cara cierre el ojo izquierdo, el derecho o los dos, que se ponga triste, alegre o incluso que bostece.

```
100 REM *****
110 REM * ANIMACION DE UNA CARA *
120 REM *****
130 REM
140 REM *** DEFINICION DE LAS DISTINTAS EXPRESIONES ***
150 REM
160 DIM A$(8)
170 REM
180 REM *** LOS DOS OJOS ABIERTOS ***
190 REM
200 LET A$(1)="*  ( )  ( )  *"
210 REM
220 REM *** OJO IZQ. CERRADO ***
230 REM
240 LET A$(2)="*  --  ( )  *"
250 REM
260 REM *** OJO DER. CERRADO ***
270 REM
280 LET A$(3)="*  ( )  --  *"
290 REM
300 REM *** LOS DOS OJOS CERRADOS ***
310 REM
320 LET A$(4)="*  --  --  *"
330 REM
340 REM *** BOCA EN POSICION NORMAL ***
350 REM
360 LET A$(5)="*  ---  *"
370 REM
380 REM *** SONRISA ***
390 REM
400 LET A$(6)="*  \-/  *"
410 REM
420 REM *** TRISTEZA ***
430 REM
440 LET A$(7)="*  /-\  *"
450 REM
460 REM *** BOSTEZANDO ***
470 REM
```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

480 LET A$(8)=" * (-) *"
490 CLS
500 REM
510 REM *** DIBUJO DE LA CABEZA ***
520 REM
530 PRINT " *****"
540 PRINT " *          *"
550 PRINT " *              *"
560 PRINT "* ( )      ( ) *"
570 PRINT "*              *"
580 PRINT "*      !      *"
590 PRINT "*              *"
600 PRINT "*              *"
610 PRINT " *          *"
620 PRINT " * ----  *"
630 PRINT " *          *"
640 PRINT " *      *"
650 PRINT " *      *"
660 PRINT
670 PRINT "S - ALEGRE      D - OJO DERC. "
680 PRINT "T - TRISTE      I - OJO IZQU. "
690 PRINT "L - DORMIR      B - BOSTEZAR"
700 REM
710 REM *** PROGRAMA PRINCIPAL ***
720 REM
730 FOR I=1 TO 45
740 NEXT I
750 LET B$=INKEY$
760 IF B$="S" OR B$="s" THEN LOCATE 10,1:PRINT A$(6):GOTO 730
770 IF B$="T" OR B$="t" THEN LOCATE 10,1:PRINT A$(7):GOTO 730
780 IF B$="B" OR B$="b" THEN LOCATE 10,1:PRINT A$(8):GOTO 730
790 IF B$="D" OR B$="d" THEN LOCATE 4,1:PRINT A$(3):GOTO 730
800 IF B$="I" OR B$="i" THEN LOCATE 4,1:PRINT A$(2):GOTO 730
810 IF B$="L" OR B$="l" THEN LOCATE 4,1:PRINT A$(4):GOTO 730
820 LOCATE 4,1:PRINT A$(1)
830 LOCATE 10,1:PRINT A$(5)
840 GOTO 730

```

Este programa nos muestra otra de las facetas de la animación. No hace falta que el objeto a animar se mueva por la pantalla. Puede ser una cara o cualquier otra figura que tenga que permanecer estática en la pantalla.

Se recomienda al lector que estudie el programa con atención hasta comprender su funcionamiento.

Las variaciones que hay que realizar en este programa para el COMMODORE, AMS-TRAD, MSX y SPECTRUM son las siguientes:

COMMODORE:

```

490 PRINT CHR$(147)
750 GET B$
760 IF B$="S" OR B$="s" THEN POKE
214,9:POKE 211,0: PRINT A$ (6): GOTO 730
770 IF B$="T" OR B$="t" THEN POKE
214,9:POKE 211,0: PRINT A$ (7): GOTO 730
780 IF B$="B" OR B$="b" THEN POKE
214,9:POKE 211,0: PRINT A$ (8): GOTO 730
790 IF B$="D" OR B$="d" THEN POKE

```

```

214,9:POKE 211,0: PRINT A$ (3): GOTO 730
800 IF B$="I" OR B$="i" THEN POKE
214,9:POKE 211,0: PRINT A$ (2): GOTO 730
810 IF B$="L" OR B$="l" THEN POKE
214,9:POKE 211,0: PRINT A$ (4): GOTO 730
820 POKE 214,3:POKE 211,0: PRINT A$ (1)
830 POKE 214,9:POKE 211,0: PRINT A$ (5)

```

SPECTRUM:

```

160 DIM A$(8,13)
760 IF B$="S" OR B$="s" THEN PRINT AT
9,0; A$(6):GOTO 730
770 IF B$="T" OR B$="t" THEN PRINT AT
9,0; A$(7):GOTO 730
780 IF B$="B" OR B$="b" THEN PRINT AT
9,0; A$(8):GOTO 730
790 IF B$="D" OR B$="d" THEN PRINT AT
3,0; A$(3):GOTO 730
800 IF B$="I" OR B$="i" THEN PRINT AT
3,0; A$(2):GOTO 730
810 IF B$="L" OR B$="l" THEN PRINT AT

```

```

3,0: A$(4):GOTO 730
      820 PRINT AT 3,0:A$(1)
      830 PRINT AT 9,0:A$(5)

```

```

1,4:PRINT A$(4):GOTO 730
      820 LOCATE 1,4:PRINT A$(1)
      830 LOCATE 1,10:PRINT A$(5)

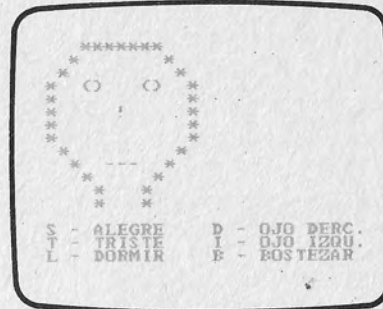
```

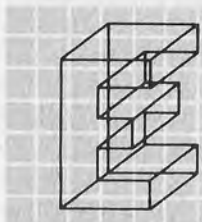
AMSTRAD y MSX:

```

760 IF B$="S" OR B$="s" THEN LOCATE
1,10:PRINT A$(6):GOTO 730
770 IF B$="T" OR B$="t" THEN LOCATE
1,10:PRINT A$(7):GOTO 730
780 IF B$="B" OR B$="b" THEN LOCATE
1,10:PRINT A$(8):GOTO 730
790 IF B$="D" OR B$="d" THEN LOCATE
1,4:PRINT A$(3):GOTO 730
800 IF B$="I" OR B$="i" THEN LOCATE
1,4:PRINT A$(2):GOTO 730
810 IF B$="L" OR B$="l" THEN LOCATE

```





N el tomo anterior vimos cómo funcionaba el programa GESTOR DE FICHEROS, pero dejamos algunas subrutinas para este tomo. Estas subrutinas son las que se incorporaron nuevas al programa y aquéllas que se va-

riaron para obtener un programa más compacto y rápido. Así, pues, las rutinas que vamos a ver en este tomo (que será el último que trate sobre el programa GESTOR DE FICHEROS) son las siguientes:

- Subrutina de introducción de fichas.
- Subrutina de ver la ficha.
- Subrutina de ordenación alfanumérica.
- Subrutina de búsqueda.

■ Subrutina de introducción de fichas

Esta es la rutina que se utiliza para introducir una ficha nueva o para modificar una ficha ya existente en el fichero. Su funcionamiento es el siguiente:

Líneas 8060 a 8080. Se imprime un mensaje que avisa al usuario de la ficha que estamos metiendo actualmente. El número de ficha que vamos a introducir viene dado por la variable numérica NF. Esta variable nos la pasa el programa principal.

Líneas 8090. En esta línea comienza un bucle dentro del cual vamos a introducir, uno a uno, todos los campos de que se compone la ficha. El bucle va desde uno (1), hasta N (1). Si recordamos, este elemento del vector N (), es el que nos da el número de campos por ficha

que tiene el fichero con el cual estamos trabajando.

Línea 8100. Colocamos el cursor en la décima línea de la pantalla.

Línea 8110. Imprimimos el nombre del campo (que se almacena en el vector M\$ ()) seguido de dos puntos y un espacio.

Línea 8120. Hacemos que la variable X tenga el valor de la columna donde se tiene que poner el cursor para la entrada de datos. Esta columna la hallamos sumando cuatro a la longitud del nombre del campo.

Línea 8130. Hacemos que la variable numérica Y tenga el valor 10. Estas dos variables, X a Y, son las que le dirán a la rutina que se encuentra a partir de la **línea 4680** (rutina de entrada de datos) en qué posición de la pantalla tiene que colocar el cursor y empezar a leer datos del teclado.

Línea 8140. Hacemos que la variable alfanumérica M\$ sea igual a "z". Esta variable, como todos recordamos, le dice a la rutina de entrada de datos cuál es el carácter con código ASCII mayor que al usuario le está permitido introducir. Cualquier carácter con código ASCII mayor que el de la letra "z" que el usuario introduzca por el teclado, no será admitido.

Línea 8150. Almacenamos en la variable alfanumérica W\$ el carácter con el código ASCII más bajo que el usuario puede introducir. Cualquier otro que tenga un código ASCII más bajo que el del número "0" no será aceptado con la excepción de la barra de espacio, el delete y la tecla ENTER (o RETURN).

Línea 8160. Almacenamos en la variable LO la longitud de dicho campo en caracteres y llamamos a la rutina de entrada de datos. La variable LO le dice a la rutina el número máximo de caracteres que el usuario puede intro-

ducir. Nosotros teníamos almacenado este dato en el vector N (), de forma que la longitud del campo número uno se encuentra en el elemento número dos del vector, el campo número dos en el elemento número tres del vector, y así sucesivamente.

Línea 8170. Una vez que el usuario ha introducido el campo y que la subrutina ha devuelto el control a esta subrutina, colocamos el cursor en la línea número 20 de la pantalla.

Línea 8180. Y preguntamos si lo que ha escrito el usuario está correcto.

Línea 8190. Leemos del teclado si el usuario ha pulsado alguna tecla. El resultado de la lectura lo almacenamos en la variable alfanumérica A\$. Si dicha variable es igual a la cadena vacía (A\$=" "), entonces es que no se pulsó ninguna tecla, con lo que nos volvemos a esta misma línea hasta que el usuario haya pulsado una tecla.

Línea 8200. Si el usuario pulsó una "N" o una "n", entonces nos volvemos a la **línea 8100** para que repita la introducción del campo del cual estamos hablando.

Línea 8210. Si el usuario no pulsó ni una "S" ni una "s", entonces nos volvemos a la **línea 8190** hasta que pulse una de las teclas permitidas.

Línea 8220. En el caso de que el usuario haya pulsado una "S" o una "s", entonces almacenamos en el vector F\$ () (vector donde se almacenan todas las fichas) el nuevo campo que acabamos de obtener. Además, también le unimos el carácter 254, (el carácter separador), pues lo siguiente que introduzcamos será otro campo.

Línea 8230. Colocamos el cursor en la **línea 10**.

Línea 8240. E imprimimos dos líneas de espacios para borrar todo lo que en ellas hubiese.

Línea 8250. Colocamos el cursor en la **línea 20**.

Línea 8260. E imprimimos una línea de espacios en blanco para borrar el mensaje de confirmación.

Línea 8270. Aquí termina el bucle. Cada vez que el programa llegue a esta línea, volveremos a la **línea 8100** para preguntar al usuario el valor del siguiente campo. Siempre que no hayamos introducido ya todos los campos.

Línea 8280. Cuando hemos introducido todos los campos de que se compone la ficha, devolvemos el control al programa principal.

■ Subrutina para ver una ficha

La siguiente subrutina se utiliza muchas veces en el programa. Su función es imprimirnos en pantalla el contenido de una ficha. Para decirle a la rutina qué ficha queremos ver, sólo tenemos que pasarle el número de ficha en la variable numérica NF.

Línea 8340. Borrarnos la pantalla y hacemos que la variable numérica CC se ponga a uno. Esta variable será la encargada de contar el número de campos de la ficha a imprimir que ya están impresos en pantalla.

Línea 8350. Imprimimos el número de ficha que estamos visualizando.

Línea 8360. Imprimimos el nombre del primer campo seguido de un espacio en blanco. Después de las comillas nos encontramos con un punto y coma (;) porque queremos que el valor de dicho campo en dicha ficha se imprima a continuación del nombre del campo.

Línea 8370. Preguntamos si el primer carácter de esta ficha tiene el código ASCII igual a 253. En caso afirmativo, esto significa que dicha ficha está borrada momentáneamente, por lo que avisamos al usuario, llamamos a la rutina que se encarga de pedirnos que pulsemos una tecla y devolvemos el control a la parte del programa que llamó a esta rutina.

Línea 8380. Si el primer carácter no tenía el código ASCII igual a 253, entonces significa que esta ficha se puede visualizar. En tal caso comenzamos un bucle desde dos (2) hasta la longitud de la ficha dentro del cual vamos a ir imprimiendo todos los campos que la componen. No comenzamos el bucle desde el primer carácter porque sabemos que éste tiene el código ASCII 254 y, por tanto, nos lo podemos saltar.

Línea 8390. Preguntamos si el carácter al cual está apuntando la variable Z dentro de la ficha tiene código ASCII 254. En caso afirmativo significa que acabamos de terminar de imprimir un campo (eso lo hacemos en la siguiente línea), con lo que, como nos encontramos en las puertas de un nuevo campo, le sumamos uno a la variable CC (Contador de Campos), imprimimos el nombre del siguiente campo y nos vamos a la línea que marca el final del bucle.

Línea 8400. Si dicho carácter no tiene un código ASCII igual a 254 significa que pertenece a un campo, con lo cual lo imprimimos. El punto y coma (;) del final sirve para que el

siguiente carácter de dicho campo se escriba a continuación.

Línea 8410. En esta línea termina el bucle. A esta línea es a la que transferimos el control del programa cuando cambiamos de campo.

Línea 8420. Llamamos a la rutina que se encarga de decirle al usuario que pulse una tecla.

Línea 8430. Devolvemos el control a la parte del programa que llamó a esta rutina.

■ Subrutina de ordenación alfanumérica

Esta rutina es la encargada de ordenar todo el fichero por el campo que diga el usuario. Esta rutina se ha modificado para conseguir más velocidad y más fiabilidad. Además ha quedado reducida a menos líneas de las que antes tenía.

Línea 6940. Borramos la pantalla.

Líneas 6950 a 7000. Se imprimen en pantalla una serie de mensajes que le dicen al usuario en qué parte del programa nos encontramos, por qué campo vamos a ordenar el fichero y que espere un momento.

Línea 7010. Preguntamos si el número de campo por el que el usuario quiere ordenar es válido, si existe. En caso afirmativo nos vamos a la **línea 7120** para empezar la ordenación.

Líneas 7020 a 7080. En caso negativo, ese campo no existe en este fichero, imprimimos una serie de mensajes de error para que el usuario sepa que dicho campo no existe.

Línea 7090. Asignamos una serie de valores a las variables X e Y y llamamos a la rutina de pulsa una tecla. Estas variables, como todos recordarán, le dicen a la rutina de pulsa una tecla, en qué posición de la pantalla tiene que imprimir el mensaje.

Línea 7100. Borramos de nuevo la pantalla.

Línea 7110. Y devolvemos el control del programa a la parte del mismo que llamó a esta rutina.

A partir de la siguiente línea es donde se realiza la ordenación del fichero. Las fichas que han sido borradas también son ordenadas.

Línea 7120. Comenzamos un bucle que va desde uno (1) al número de fichas que actualmente tiene el fichero. Dentro de este bucle se producirá la ordenación del mismo. Por

otro lado, también en esta línea se inicializan dos variables a cero. Cada vez que el bucle dé una vuelta, dichas variables se volverán a poner a cero. La función que van a realizar la veremos un poco mas adelante.

Línea 7130. Comenzamos un segundo bucle que nos permitirá junto con el primero, ordenar todas las fichas. También comenzamos otro bucle, dentro del cual vamos a encontrar en qué posición empieza el campo por el que queremos ordenar el fichero en la ficha que estamos trabajando actualmente. Lo que hacemos a continuación es ir contando el número de caracteres que tienen código ASCII 254, para que cuando hayamos contado NC caracteres 254 saber que nos encontramos en el campo NC. Una vez que lo sabemos asignamos a la variable Z1 la posición de dicho campo en F\$ (Z). Como se puede apreciar, en esta línea utilizamos la variable CC, que es la Contadora de Campos.

Línea 7140. En esta línea, aparte de terminar el último de los bucles y de volver a inicializar la variable CC a cero, comienza otro bucle, dentro del cual vamos a buscar el campo NC dentro de F\$ (X)). Una vez que lo encontremos ambos registros (F\$ (Z) y F\$ (X)) para saber cuál es el que tiene que colocarse antes en la ordenación. En esta línea, después del bucle, realizamos lo mismo que en la anterior. Almacenamos la posición del campo NC en el registro F\$ (X) dentro de la variable Z2.

Línea 7150. Terminamos el bucle anterior y miramos cuál de los dos registros es menor. Ser menor significa que se tiene que encontrar detrás del otro. Si resulta que el registro F\$ (Z) es menor que F\$ (X) entonces intercambiamos las posiciones de ambos en el fichero. Para ello utilizamos una variable auxiliar, A\$.

Línea 7160. Aquí termina el bucle que tiene como variable índice la X.

Línea 7170. Aquí termina el bucle principal. El que tiene como variable índice a la Z.

Línea 7180. Una vez que se ha terminado el bucle, el fichero ya está ordenado. Por lo tanto, borramos la pantalla y avisamos al usuario de que la operación está terminada.

Líneas 7190 a 7210. Aparte de subrayar el mensaje anterior, se imprime otro mensaje que nos informa de que el fichero está ordenado y por qué campo lo está.

Línea 7220. Asignamos una serie de valores a las variables X e Y y llamamos a la rutina, que le dice al usuario que pulse una te-

TRUCOS Y RUTINAS BASICAS

cla. Estas dos variables (X e Y) le dicen a la rutina en qué posición de la pantalla se tiene que colocar el mensaje PULSA UNA TECLA.

Línea 7230. Borrarnos la pantalla de nuevo.

Línea 7240. Y devolvemos el control del programa a la parte principal del mismo.

Por fin, y ya para terminar con este programa, sólo nos queda por ver la rutina de búsqueda de fichas. Esta rutina también se modificó para hacer que el programa fuese mucho más veloz. Esta rutina hace uso de la rutina de presentación de fichas en pantalla. Su funcionamiento es el siguiente:

Dentro de una rutina, nos encontramos con que podíamos buscar una, o varias, fichas por su número dentro del fichero, porque tuviesen un cierto valor en un cierto campo, o ver todas aquellas fichas que han sido borradas.

Las líneas que van desde la 6280 hasta la 6390 se encargan de borrar la pantalla; y de imprimir una serie de mensajes, en pantalla. Entre ellos destaca el que le dice al usuario qué tipo de búsqueda vamos a realizar.

Línea 6400. En esta línea hacemos una llamada a otra de las partes de esta rutina. La parte elegida depende del valor de la variable SW, que es la que nos indica qué tipo de visualización desea el usuario.

Línea 6410. Cuando la visualización ha terminado, el programa vuelve a esta línea. Borrarnos la pantalla.

Líneas 6420 a 6450. Se avisa al usuario de que se ha terminado la visualización, que no se han encontrado más fichas con las características pedidas.

Línea 6460. Asignamos a la variable X el valor uno (1) y a la variable Y el valor diez (10). Estas dos variables son utilizadas por la rutina de pulsa una tecla.

Línea 6470. Llamamos a la rutina de pulsa una tecla.

Línea 6480. Y, una vez que el usuario a pulsado una tecla, devolvemos el control al programa principal.

El grupo de líneas que aparece a continuación es el encargado de realizar la visualización de la fichas cuya posición numérica dentro del fichero esté comprendida entre dos números dados por el usuario.

Línea 6520. Comienza un bucle desde la primera ficha a visualizar (N1) y la última (N2).

Línea 6530. Hacemos que NF sea igual a R., R es la variable índice del bucle y va to-

mando todos los valores entre N1 y N2. NF (Número de Ficha) es la variable que, como ya hemos visto, necesita la rutina de visualización para mostrarnos una ficha por pantalla.

Línea 6540. Llamamos a la rutina de visualización.

Línea 6550. Borrarnos la pantalla antes de imprimir la siguiente ficha.

Línea 6560. Aquí termina el bucle.

Línea 6570. Y devolvemos el control a la rutina general para que nos diga que la operación está terminada.

Las líneas que aparecen a continuación se encargan de mostrarnos en pantalla todas las fichas que en un cierto campo tengan un valor determinado. Se entiende por valor cualquier número o secuencia de caracteres. Hay que decir que esta rutina sólo mira si dicha secuencia se encuentra al principio del campo y no en medio de éste. Lo bueno que tiene es que podemos decirle al programa que, por ejemplo, nos busque todas las fichas que en el campo número 2 tengan una "A". Cuando el ordenador empiece a buscar, nos mostrará por pantalla todas aquellas fichas en las que el campo número 2 empieza por "A".

Línea 6610. Preguntamos si el fichero está definido. En caso afirmativo, nos vamos a la **línea 6690**.

Línea 6620 a 6650. Si el fichero no está definido, imprimimos un mensaje de error y avisamos al usuario que el campo por el cual quiere buscar no existe.

Línea 6660. Llamamos a la rutina de pulsa una tecla.

Línea 6670. Borrarnos la pantalla.

Línea 6680. Y devolvemos el control al principio de la rutina.

Línea 6690. La serie de caracteres a buscar se encuentra almacenada en la variable A\$. Lo primero que hacemos en esta línea es asignar a la variable alfanumérica Jñ el valor A\$. También comenzamos un bucle desde 1 hasta TT (el número de fichas que tiene el fichero) dentro del cual vamos a mirar, ficha a ficha, si reúnen las condiciones necesarias para ser visualizadas. También ponemos a cero una serie de variables que utilizaremos en el transcurso del bucle.

Línea 6700. Comenzamos un bucle. Dentro de éste lo que haremos será buscar la posición donde empieza el campo NC dentro de la ficha que en este momento estamos tratando. Lo hacemos como ya hemos visto anteriormente. Contamos el número de caracteres con dicho código ASCII significa que nos encon-

tramos al principio del campo NC. Dicha posición se la asignamos a la variable Z1 y nos salimos del bucle.

Línea 6710. En esta línea, aparte de terminar el bucle anterior, comenzamos uno nuevo. La función de éste va a ser comparar, carácter a carácter, el contenido del campo NC con el contenido de J\$. Si todos los caracteres, hasta la longitud de J\$, coinciden, entonces podemos imprimir la ficha en pantalla.

Línea 6720. Aquí termina el último bucle que hemos abierto.

Línea 6730. Si CR (variable que cuenta el número de caracteres que coinciden) es igual a la longitud de J\$, entonces significa que dicho campo es igual que J\$. Por lo tanto, hacemos que NF sea igual a Z y llamamos a la rutina de visualización de fichas. También asignamos a la variable VV el valor de Z, antes de mandarlo a la rutina de visualización, y asignamos a Z el valor de VV una vez que ha vuelto de la rutina. Esto es necesario porque la rutina de visualización utiliza también la variable Z en sus bucles y así la salvaguardamos.

Línea 6740. Terminamos el bucle principal.

Línea 6750. Y devolvemos el control a la parte inicial de la rutina.

Por fin, y ya para terminar, aparece el grupo de líneas que se encargan de buscar todas las fichas que han sido borradas.

Línea 6800. Comienza un bucle desde uno (1) hasta el número de fichas que tiene el fichero. Dentro de este bucle miraremos una

a una si las fichas están borradas. Aquéllas que lo estén las imprimiremos en pantalla.

Línea 6810. Preguntamos si el primer carácter de la ficha que estamos tratando tiene un código ASCII igual a 253. En caso negativo nos vamos a la **línea 6870** para continuar con el bucle.

Línea 6820. Si el primer carácter sí tenía código 253, entonces le quitamos dicho primer carácter a la ficha. Esto es necesario hacerlo, ya que la rutina de visualización tiene una protección para no imprimir todas aquellas fichas que estén borradas. Una vez que la ficha haya sido visualizada, volveremos a ponerle el carácter 253 al principio.

Línea 6830. Asignamos a NF el número de la ficha en curso.

Línea 6840. Llamamos a la rutina de visualización de fichas.

Línea 6850. Borrados la pantalla y asignamos a la variable Z el valor de NF. Esto es necesario porque la rutina de visualización varía el valor de la variable Z, pero no el de NF.

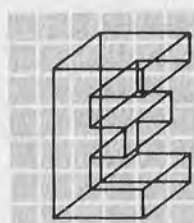
Línea 6860. Volveremos a colocar el carácter 253 al principio de la ficha que acabamos de visualizar.

Línea 6870. Y terminamos el bucle.

Línea 6880. Devolvemos el control del programa a la parte principal de la rutina.

Con esto ya podemos considerar que hemos terminado la creación del programa GESTOR DE FICHEROS. Más adelante, en tomos sucesivos, volveremos a ver más cosas referidas a la creación de ficheros, su ordenación y su impresión.

■ Arquitecturas de microprocesadores. Evolución



L núcleo fundamental de las máquinas utilizadas actualmente en informática, en muchos de los equipos electrónicos domésticos y en la mayoría de los dispositivos de control industrial, es algún tipo de microprocesa-

dor. La necesidad de aumentar las prestaciones de los equipos, junto con la posibilidad de introducir los resultados de las investigaciones en tecnología en mayor número de aplicaciones, ha forzado la evolución de estos componentes en múltiples direcciones.

Vamos a describir de manera general los pasos que la industria electrónica ha seguido en el desarrollo de los microprocesadores, dando algunos detalles particulares de algunos "especímenes" que se han distinguido a lo largo de esta acelerada carrera evolutiva. Es imposible describir todos y cada uno de los resultados que han visto la luz de forma comercial, teniendo además presente que, aunque esto se hiciera, quedaría incompleta la historia, pues ha habido muchos componentes que nunca han sido descritos en literatura pública y que esperan el paso de los años para que los investigadores los descubran, como las galerías principales de la pirámide de Keops esperan al descubridor de su secreto.

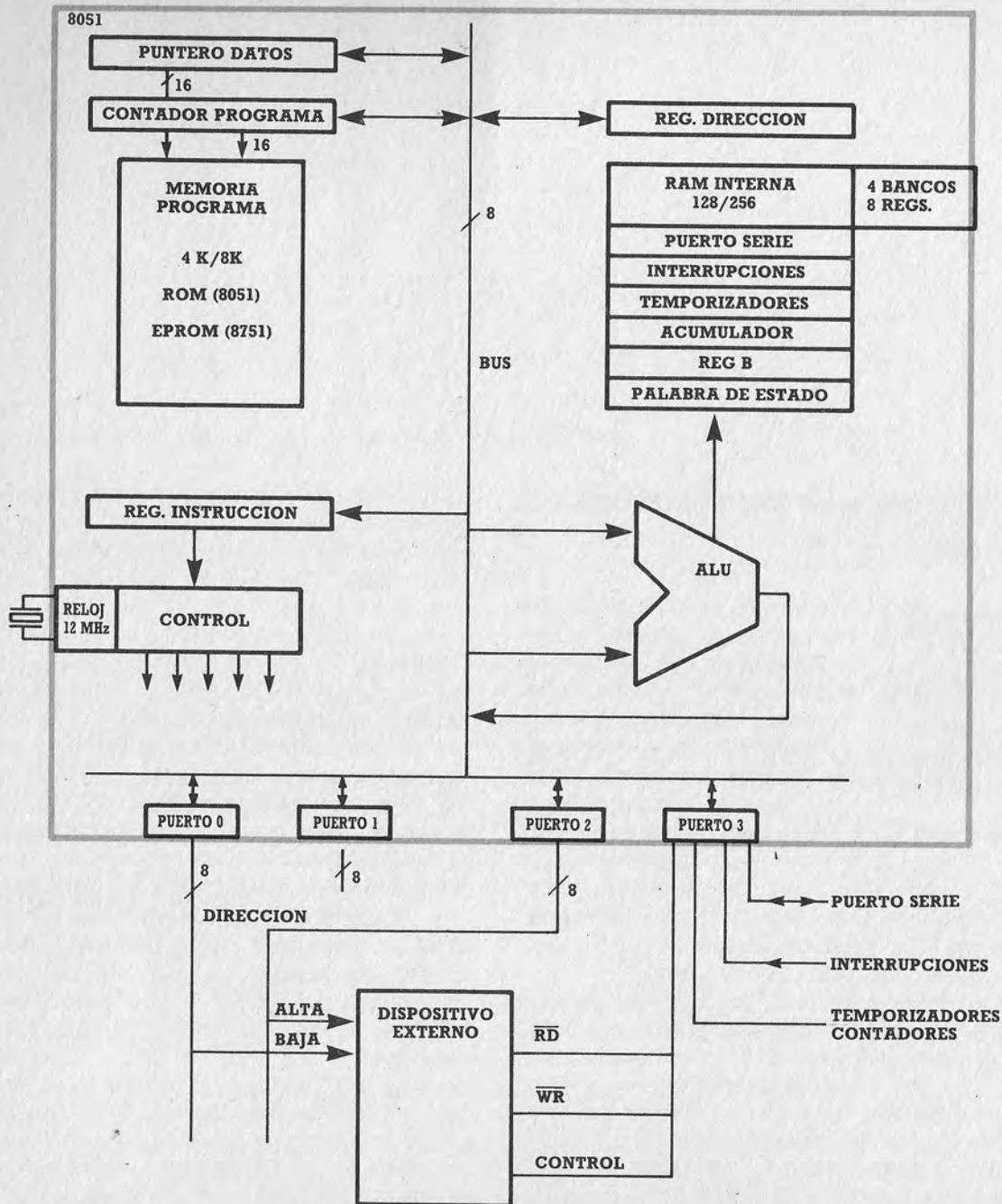
En primer lugar, veamos las posibles fuerzas que encaminan la evolución. Con ello podremos situarnos en nuestras propias coordenadas y hacernos idea de cómo pueden evolucionar en un futuro próximo, si conoce-

mos, asimismo, las fuerzas que impulsan esta evolución.

En cuanto a las posibilidades de mejora tecnológica, por reducción de las dimensiones del microcircuito, se han ido escalonadamente consiguiendo circuitos con anchos de pista desde las 10 micras hasta el estado actual inferior a la micra. La reducción en dimensiones afecta igualmente al consumo y a la velocidad de proceso obtenible en el circuito. También la reducción de tamaño permite la inclusión en el mismo "empaquetamiento" de más componentes, con la ventaja consiguiente de reducción del número de conexiones externas y los problemas asociados a las conexiones.

Otra posibilidad de mejora es la utilización de materiales diferentes que permitan resolver los problemas del silicio. Versiones avanzadas de NMOS y CMOS han permitido mejorar las velocidades y reducir los consumos, para poder juntar hasta un número de puertas por encima del millón en un solo chip, con rendimientos elevados de producción. Otros semiconductores, como el GaAs, está dando buenos resultados, pero sólo están disponibles en aplicaciones avanzadas, donde sus elevadas prestaciones puede compensar el elevado coste.

Pero es en el área de la arquitectura del microprocesador donde la evolución es más espectacular. Se ha pasado de componentes de propósito general, con necesidad de muchos componentes auxiliares, a una gran variedad de gamas de productos perfectamente adaptados a los problemas a resolver, con posibilidades de crecimiento prácticamente ilimitado dentro de un campo determinado. La manera más gráfica de expresar esta evolución es mediante un bosque en el que los ár-



boles representan las diferentes familias, que hincan sus raíces en las diferentes tecnologías y cuyos frutos representan los diferentes microprocesadores, con diferente forma según el número de bits de su arquitectura. Pueden apreciarse ramas desgajadas de algunos árboles que han dado sabrosos frutos.

Veamos la evolución histórica de las diferentes familias de microprocesadores. En los recuadros específicos se describen las características principales de algunos de ellos y se muestra, en esquema, la conexión entre sus

unidades internas, para comprender el funcionamiento.

La historia viene de no muy lejos, cuando, en 1970, un grupo de ingenieros de diseño de circuitos integrados propone la creación de una unidad que realizara lo que hasta ese momento requería, al menos, media docena de circuitos diferentes. El estado del arte de integración en aquellos momentos no permitía obtener elevados rendimientos de fabricación si el número de puertas integradas sobrepasaba el millar. Al no encontrar respuesta posi-

reccionamiento y mejoras en la comunicación con los periféricos, al crearse componentes específicos para todas las funciones necesarias: memorias, comunicaciones, discos, temporizadores, control de interrupciones, control de acceso directo a memoria.

A esta generación pertenecen los Intel-8080, Motorola-6800, Fairchild-F8, Rockwell-6502, Signetics-2650, todos ellos de gran éxito comercial en aplicaciones industriales y con numerosas versiones en diferentes tecnologías integradas y proporcionados por varias segundas fuentes, además de la casa original.

Todos ellos poseen bus de datos de ocho bits, bus de direcciones de 16 bits, con lo que pueden direccionar 64K y una o dos entradas para indicación de interrupción.

El empaquetamiento es en circuitos de 40 patillas, excepto el F8, que se realizó sobre dos circuitos separados. Requieren alimentación positiva y negativa, excepto el 6800, que sólo requiere +5V.

Continúa la evolución con dos direcciones principales:

1. Aumento del número de instrucciones, manteniendo la arquitectura básica.
2. Aumento del tamaño de la superficie de circuito integrado, incluyendo funciones adicionales, como memoria, puertos de entrada/salida, reloj, temporizadores, etc., dando paso al microcomputador en un chip.

De la primera línea evolutiva se derivan micros del tipo del Z80, que es una variante avanzada del 8080. También el aumento de velocidad hace que las prestaciones, en general, resulten incrementadas.

De la segunda línea se derivan los del tipo 8085, 6801, 6805, que se han utilizado por millares como microcontroladores de todo tipo de periféricos. Las versiones actuales del tipo 8048, 8051 incluyen dispositivos adicionales para optimizar su empleo en control de periféricos en tiempo real, como es el juego de instrucciones sobre datos de un solo bit, registros de desplazamiento, temporizadores programables, etc.

Hay una familia de micros de un solo bit que, aunque ha sido aplicada en la industria, no ha tenido sucesores: es el Motorola 14500. Este curioso micro permite la realización de bucles de control, en procesos industriales no muy rápidos, con suma sencillez y economía de medios. Actualmente está totalmente superado por los microcomputadores en un chip.

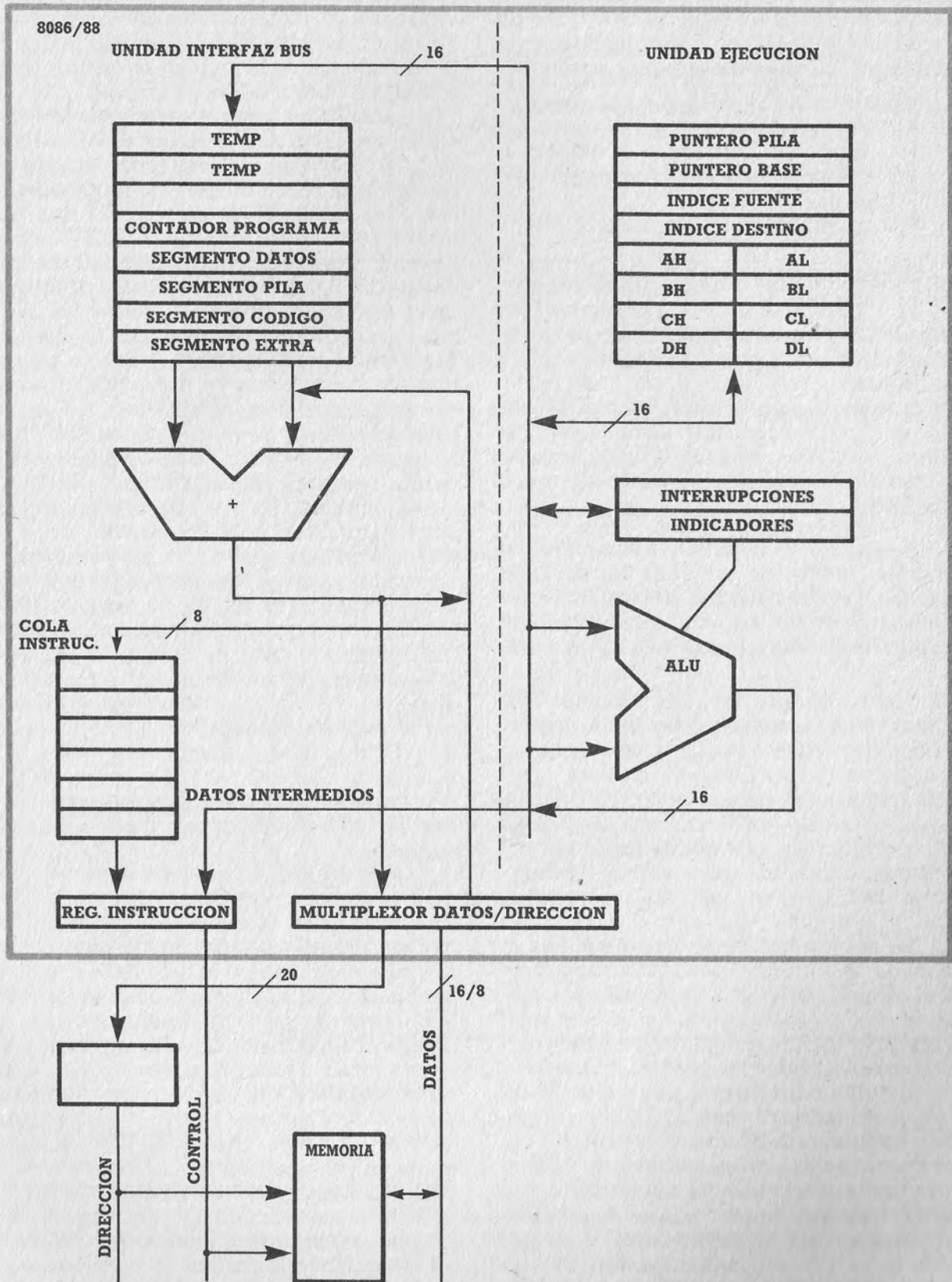
El siguiente salto lo constituye el paso a bus de datos de 16 bits. El salto es muy grande y el aumento de complejidad de fabricación es sólo la punta del iceberg que constituye el problema. Por eso hay un período en el cual se utiliza arquitectura interna de 16 bits con conexión al resto del micro con bus de ocho bits. Esta transición permite utilizar todo el conjunto de programas de periféricos, desarrollado para los micros antiguos, ganando en prestaciones con las nuevas máquinas. Al mismo tiempo, la velocidad de CPU y memoria se continúa incrementando por la mejora de la tecnología.

A esta época pertenecen los micros 8088 de la línea del 8080, pero compatible con 8086. Poseen juego de instrucciones enriquecido con multiplicación y división entera y utilizan la técnica de segmentación para poder direccionar hasta un Mega. En gran parte, su éxito comercial ha sido causado por la adopción de este micro, en el año 1981, por IBM, para el lanzamiento del ordenador personal. En el momento de su aparición existía ya en el mercado otro micro, derivado de la otra familia de gran éxito de Motorola, el 68000, con arquitectura más regular y estructurada, que ha tenido también sucesores con bus de conexión externa reducido, como el 68008.

El paso de ocho a 16 bits se hace de manera gradual en la mayoría de fabricantes, pero teniendo como objetivo, a corto plazo, el alcanzar los micros de 32 bits para competir en el área de los minis. Para poder realizar funciones avanzadas es frecuente que las familias posean un miembro "listo" que pueda conectarse opcionalmente, para mejorar las prestaciones en un campo determinado. Por ejemplo, para la realización de operaciones numéricas, la familia del 8086 y sucesores poseen el 8087 y sucesores, que permiten mejorar el rendimiento en coma flotante y manejo en series de caracteres. Para ello estos micros especiales tienen internamente una estructura idónea para realizar las operaciones con registros de mayor longitud que el micro básico y una unidad aritmética-lógica especialmente diseñada para ese tipo de operaciones. Además pueden trabajar en paralelo con la CPU base, una vez que han recibido los datos sobre los que operar.

Los microcomputadores en un chip también evolucionan hacia los 16 bits, presentando, como sucesores de la familia del 8051, el 8096, que, además, incluye entradas y salidas analógicas.

8086/88



EL TALLER DEL HARDWARE

En los movidos años 80 se plantea una fuerte discusión entre los diseñadores sobre las líneas más convenientes de desarrollo de los micros, con objeto de cubrir óptimamente varios objetivos que parecen enfrentados:

- Máximas prestaciones en ejecución.
- Facilidad de desarrollo de aplicaciones.
- Rendimiento en fabricación del componente.
- Regularidad de la estructura.

Se plantean dos líneas de diseño:

1. CISC = Complex Instruction Set Computer. Micros dotados de un juego de instrucciones amplio y potente, aunque requiera que las instrucciones tengan tiempo de ejecución variable. Mucho espacio de circuito utilizado en el decodificador y en la unidad de control. Lógica de ejecución compleja con algún tipo de paralelismo para paliar la lentitud (pipe-lining).

2. ARISC = Reduced Instruction Set Computer. Micros de juego de instrucciones reducido, pero que pueden ejecutarse de forma similar, y en general, en un solo ciclo. Decodificación simple. Lógica ejecución sencilla.

Con la primera línea de diseño es fácil de desarrollar los sistemas operativos que demandan las aplicaciones actuales, obtienen prestaciones razonables, pero se tiene constancia que muchas partes de la circuitería interna no se utilizan jamás. Es decir, se paga un alto precio por algo que podría realizarse por programa en aquellos casos en que fuera necesario. Hay, pues, una mala utilización del silicio.

Por el contrario, con la segunda línea se obtiene un gran rendimiento en todos los casos, el código puede optimizarse casi siempre, y en el caso de aplicaciones especiales, puede añadirse algún circuito de propósito particular para resolver el problema concreto.

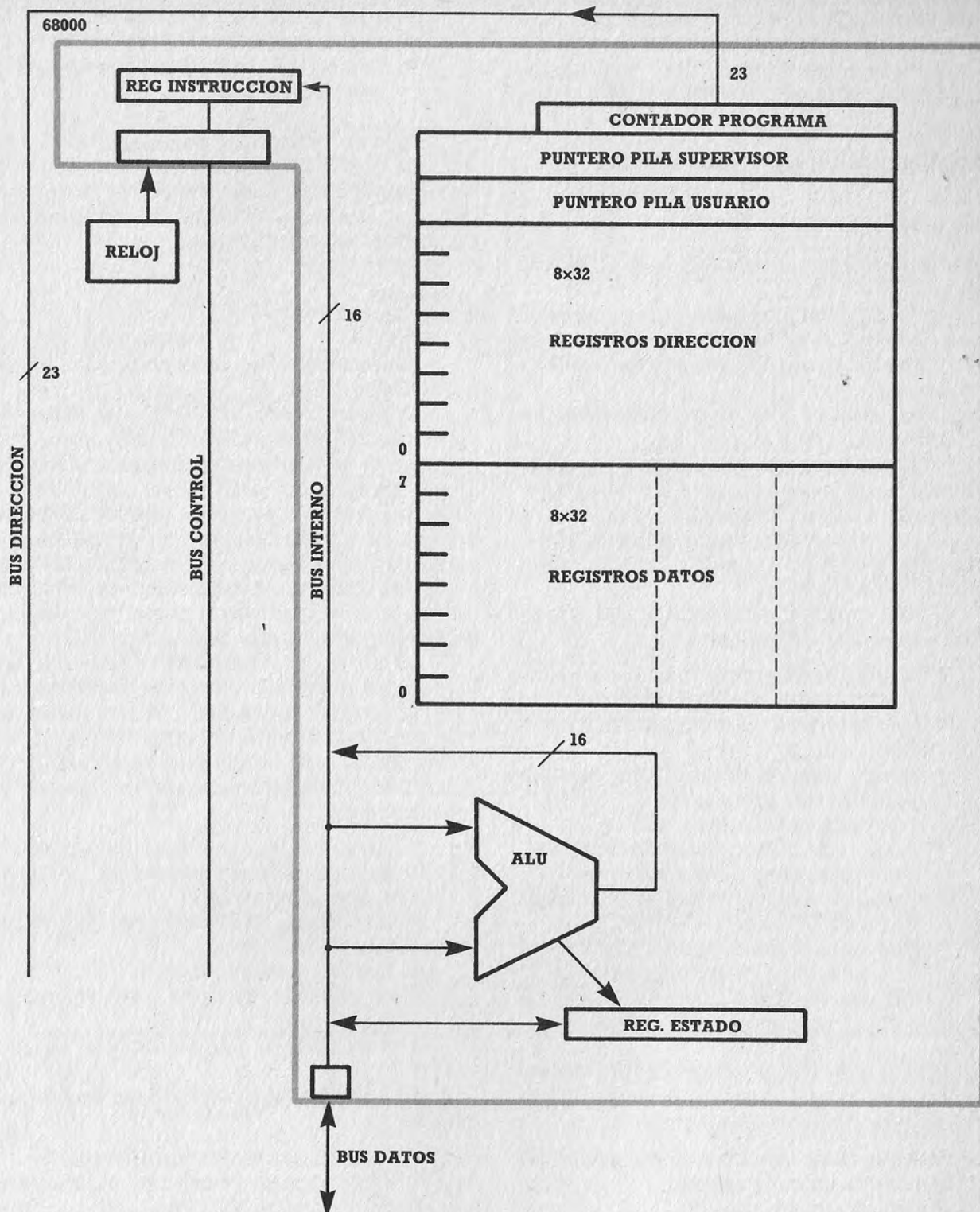
El hecho constatado es que ambas líneas no son totalmente contrapuestas y siguen apareciendo sucesores de ambas filosofías y parece que van a seguir apareciendo en el futuro. Una pequeña indicación de las áreas que cubrirá cada uno de los tipos es: la primera, para equipos de propósito general y sucesores de minis, y la segunda, para máquinas de propósito específico y estaciones de trabajo orientadas a un problema concreto.

Como elemento de la primera línea puede destacarse el 80386, que es sucesor del tradicional 8080, incorporando funciones propias de los minis de no muchos años atrás, como: protección de memoria, bus de datos y direcciones de 32 bits, memoria virtual.

Como elemento destacado de la segunda línea es digno de mención el IMS-T414 de INMOS, también denominado Transputer (TRANSistor comPUTER), que presenta un cambio radical en la arquitectura que realmente resulta revolucionario. Está diseñado teniendo presente su conectabilidad con otros elementos iguales a él. Para ello, además de una arquitectura casi RISC, posee cuatro canales de comunicación asíncrona que pueden funcionar simultáneamente y que le permite distribuir la información necesaria entre los elementos que componen la red, de manera casi instantánea, pues la velocidad de transmisión puede alcanzar los 20 Megabits por segundo. Según las especificaciones del componente, utilizando las estructuras propuestas, la potencia puede crecer linealmente con el número de unidades utilizadas, sin presentar saturaciones, como es frecuente encontrar en los multiprocesadores utilizados hasta la fecha. Un elemento digno de consideración es el lenguaje utilizado para el desarrollo utilizando Transputers: Occam. Es un nuevo tipo de lenguaje orientado a procesos en que componen la red de Transputers.

Como ramas paralelas, con menor volumen de producción, pero igualmente de gran interés estructural y práctico, es necesario incluir familias de micros de propósito más particular:

- Procesadores de señal. Generalmente para el proceso de datos que se originan en algún tipo de sensor: acústico, imagen, radar, etc. Presentan como característica general que hay que procesar el dato recién recibido mediante operaciones de multiplicación y suma, para dar como resultados otro dato, generalmente del mismo tipo. Es muy común que su estructura interna esté compuesta de memoria separada de datos y de programas, para optimizar el tiempo de ejecución (PD7720A, UDPI01, TMS32030, DSP-128). Dentro de los procesadores de señales existen micros diseñados para el proceso de señales analógicas directamente. Hasta el presente ha habido un número muy limitado de componentes de estas características, diseñados para aplicaciones de control y filtrado de señales analógicas (Intel 2920).



• Micros "en porciones", para permitir el diseño con el ancho apropiado a la aplicación. Permiten simular cualquier estructura o juego de instrucciones mediante el número apropiado de chips y el microprograma adecuado (AMD2900).

Es necesario mencionar otra tendencia actual y que avanza con aceleración creciente: la creación de micros de propósito específico a partir de circuitos formados por conjuntos de funciones previamente integradas sobre la oblea y que sólo esperan a ser conec-

EL TALLER DEL HARDWARE

tadas en su capa superior, o a partir de librerías de módulos normalizados, la generación de micros totalmente específicos y optimizados para una aplicación determinada.

■ Retrato robot de varios micros

Intel 8051

Microprocesador de ocho bits en tecnología NMOS y CMOS.

Es una familia que cubre diferentes versiones que varían en la cantidad de memoria y en el tipo de memoria, que puede ser interna o externa.

Diseñado por Intel tiene numerosos fabricantes como segunda fuente.

Su precio es muy bajo (unos cinco dólares), salvo para la versión con EPROM, que resulta apropiada para prototipos (50 dólares).

Es un microcomputador autosuficiente, con posibilidades de expansión de memoria y conexión a periféricos.

Como características principales de su arquitectura merece citarse:

- Direccionamiento regular hasta 128K.
- Cuatro juegos de ocho registros.
- 128/256 bytes de memoria RAM en el chip.
- Cuatro puertos de ocho bits, reconfigurables por programa.
- Tres temporalizadores/contadores.
- Fuentes de interrupción: dos externas, temporizadores y comunicación.
- Canal de comunicaciones asíncronas de hasta 500 Kbps.
- Tiempo de ejecución de instrucciones de un microsegundo para reloj de 12 MHz (salvo para MUL y DIV, que es de cuatro microsegundos).

El juego de instrucciones es muy potente y variado, considerando el precio del micro, con las características siguientes:

- Aritmética de coma fija incluyendo multiplicación y división.
- Manipulación de bits muy potente (procesador booleano).
- Direccionamiento directo, inmediato e indirecto.
- Búsqueda en tabla mediante puntero de datos.
- Saltos condicionales según indicadores de estado y comparaciones.

- Interrupciones vectorizadas para todas las fuentes.
- Empleo de la pila para paso de parámetros en subrutinas.

Existen programas para soporte de desarrollo que incluyen ensamblador, compiladores de PL/M y Basic, simulador y depurador, que pueden correrse en ordenadores personales de tipo IBM-PC.

Zilog Z80

Microprocesador de ocho bits en tecnología NMOS y CMOS.

Es un micro de extraordinario éxito por su utilización en ordenadores personales. Diseñado por Zilog (empresa formada por ingenieros, puntero de Intel), como evolución del 8085, incluyendo mejoras de circuitería y facilidades de programación, es compatible con los circuitos de soporte de la familia 8080 de Intel. Sin embargo, posee un juego completo de circuitos propios de soporte, que utilizan todas las posibilidades internas del micro.

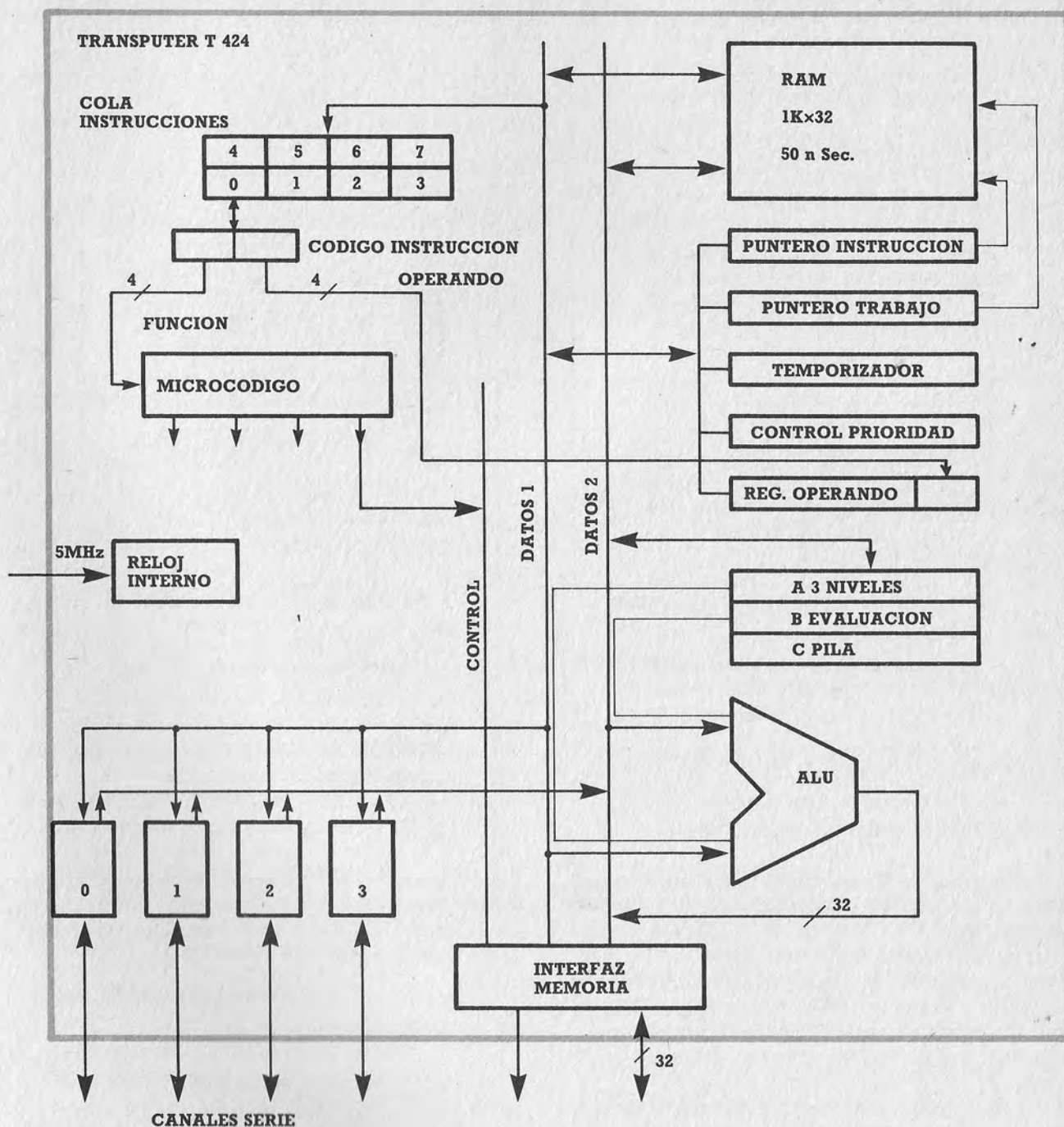
Su precio ha alcanzado valores por debajo de un dólar en volúmenes elevados por los numerosos fabricantes que compiten en este mercado, alcanzándose la cifra de 20 millones de circuitos fabricados en un año.

Como características estructurales merece citarse:

- Juego de instrucciones muy potente, incluyendo movimientos de memoria-memoria en bloque.
- Direccionamiento de 64K (su punto más flaco).
- Dos juegos de registros.
- Circuito de refresco para memorias dinámicas.
- Versiones de reloj hasta ocho MHz.

En cuanto a las posibilidades de programación:

- Aritmética de ocho bits completa.
- Aritmética de 16 bits, sólo ADD y SUB.
- Dos registros índices.
- Movimientos de memoria-memoria en bloque.
- Puntero de pila.
- Salto de programa.
- Tres tipos de funcionamiento de interrupción.
- Siete indicadores.



Posee una extensa gama de programas de desarrollo que comprende ensambladores, compiladores en todos los lenguajes y depuradores.

Intel 8088

Micro de ocho bits en tecnología NMOS y CMOS.

Micro nacido también de la evolución de la familia 8080/85, manteniendo un alto grado

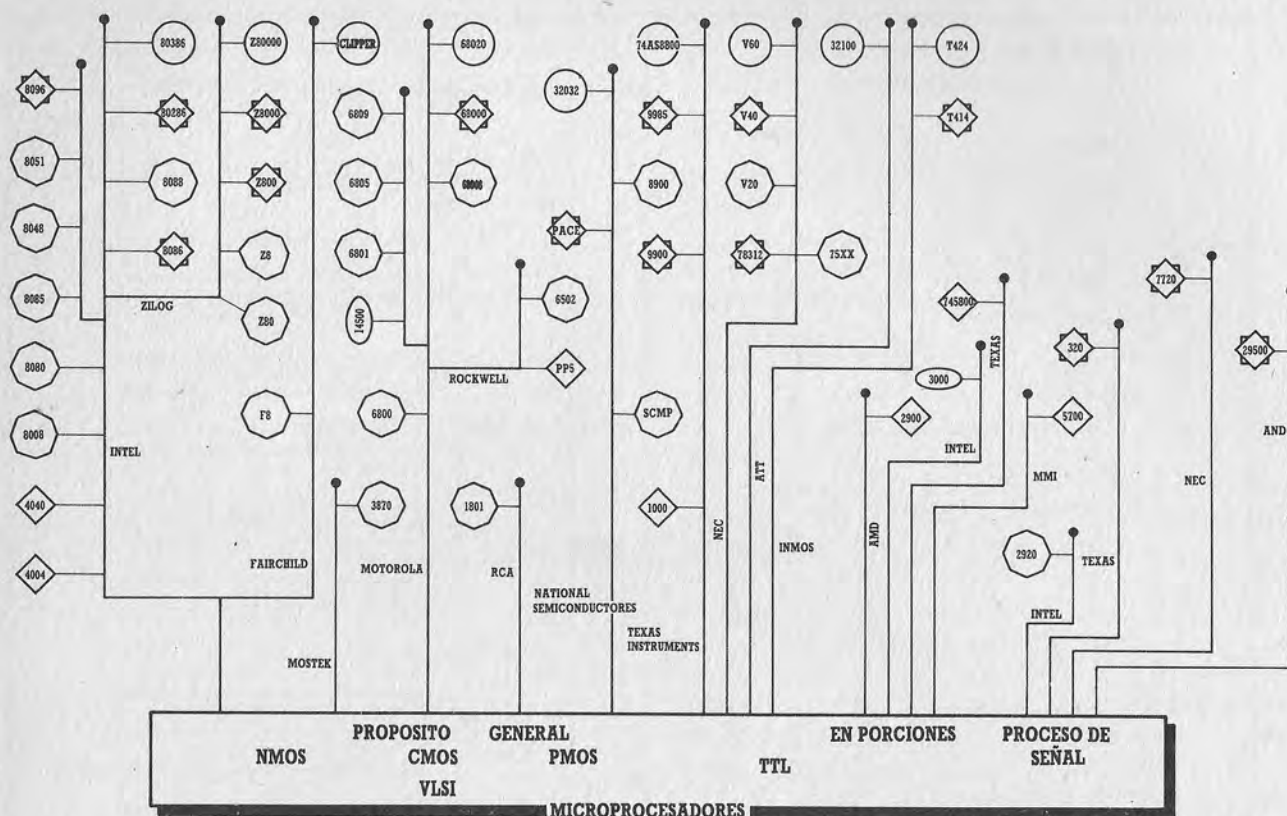
de compatibilidad con sus predecesores. El 8088 es una versión con bus de ocho bits del 8086.

Su éxito le viene de haber sido incluido en el IBM-PC. El precio actual está por debajo de los diez dólares, para compra en volumen.

Como características estructurales merece citarse:

- Juego de registros de 8/16 bits.
- Capacidad de direccionamiento hasta

EL TALLER DEL HARDWARE



un Megabyte, mediante segmentación.

- Velocidad de reloj hasta 12 MHz.
- Pila interna de instrucciones.

Dentro de los circuitos de soporte merece la pena citarse el coprocesador matemático, que permite acelerar la ejecución de instrucciones de coma flotante hasta unas 100 veces. Se conecta al microprocesador como un periférico interno, con el cual intercambia datos a través de secuencias de escape, cumplimentadas con interrupciones no enmascarables.

En cuanto a las posibilidades de programación hay que citar:

- Aritmética de 8/16 bits completa, incluyendo MUL y DIV.
- Movimiento de bloques de memoria.
- Direccionamientos directo, inmediato, indexado, relativo.
- Interrupciones hard y soft vectorizadas, utilizando circuito externo.
- Saltos condicionales dentro de +/- 128 bytes.
- Saltos dentro del segmentos auto-relativos.

- Control de iteraciones mediante instrucciones de repetición.
- Juego de indicadores de estado, sobre los que se puede condicionar saltos.

Dispone de numerosos programas de desarrollo: ensambladores, compiladores, depuradores y toda clase de herramientas, utilizando equipos de tipo IBM-PC.

Motorola 68000

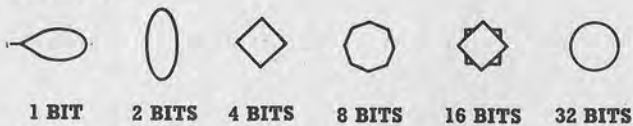
Microprocesador de 16/32 bits en tecnología NMOS y CMOS.

Diseñado por Motorola, es una familia muy variada con diferentes versiones de velocidad y elementos internos variables. Hay numerosas segundas fuentes que garantizan precios del orden de 10 dólares para la versión de 8 MHz. El micro más alto de la gama 68020 es el único en realidad de 32 bits, alcanzando un precio cercano a los 300 dólares.

La arquitectura del 68000 es muy regular, como puede verse de la descripción de sus características:

- 16 registros de 32 bits de uso general.

SIGNIFICADO



- 24 bytes de direccionamiento (32 en 68020).
- 256 bytes de memoria interna de instrucciones de alta velocidad.
- Tres niveles de pipeline.
- 18 modos de direccionamiento.
- Soporta memoria virtual.

Las posibilidades de programación son las correspondientes a un verdadero mini:

- Aritmética de 32 bits en coma fija completa.
- Los registros generales pueden usarse como índices o punteros de stack.
- Todo tipo de saltos condicionales e incondicionales.
- Siete niveles de interrupción, con 256 posibles vectores.
- 16 bits de indicadores de estado.
- Posibilidad de ejecución paso-paso y parada en una instrucción.
- Posibilidad de empleo de instrucciones privilegiadas, para distinguir estado supervisor y programa.
- Capacidad de trabajo en multiproceso.

Están disponibles equipos de desarrollo basados en ordenadores personales o en máquinas de Motorola de tipo Exormacs.

Inmos T414 Transputer

Microprocesador de 32 bits en tecnología CMOS.

Diseñado con una arquitectura innovadora, presenta características altamente atractivas para nuevos diseños.

Lanzado recientemente al mercado está esperando por una aplicación espectacular, que muestre sus muchas posibilidades. Su precio está relativamente alto, del orden de 500 dólares. Pretende ser la alternativa a los supercomputers, mediante su posibilidad de conexión en malla.

Como características principales presenta:

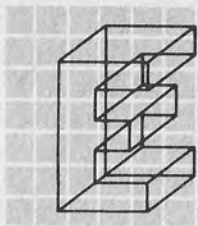
- Bus de datos y de direccionamiento de 32 bits.
- Reloj generado internamente a partir de una señal externa de referencia de 5 MHz.
- Cuatro canales de comunicación asíncrona que pueden trabajar simultáneamente hasta a 20 Mbps.
- Memoria local de dos Kbytes de 50 nS. de tiempo de acceso.
- Utilización de la memoria local como juego de registros.
- Interfaz de memoria programable, puede direccionar 4 Gbytes.
- Utiliza preferentemente direccionamiento implícito, con códigos de operación de ocho bits.
- La mayoría de las instrucciones se ejecutan en un ciclo, por lo que la media de instrucciones por segundo puede alcanzar los 10 MIPS.
- Atención inmediata a las interrupciones.
- Almacenamiento del estado de la máquina en el cambio de contexto, al producirse la interrupción.
- Soporte de operaciones en tiempo real y multitarea.

En cuanto a las posibilidades de programación:

- Aritmética de 32 bits en coma fija completa.
- Orientado a ser programado en lenguaje de alto nivel de tipo concurrente. El Occam está especialmente adaptado a esta máquina.
- Movimientos locales relativos al puntero local.
- Direccionamiento inmediato es lento, pues requiere la carga de registros de cuatro en cuatro bits.
- Acceso a los canales de comunicación a través de DMA.
- Saltos de programa relativos y condicionales.
- Permite la creación/eliminación de procesos concurrentes.
- Permite la asignación dinámica de tiempo a los procesos.
- Posibilidad de carga inicial de programa desde otro micro, a través de la red formada por ellos.

NATURALEZA Y TECNOLOGÍA

■ Biología: cadenas y pirámides alimenticias



NTRE el conjunto de animales y vegetales y el área natural que habitan existe una serie de relaciones complejas. De éstas podemos destacar las que nos permiten conocer los intercambios de materia y energía.

El flujo de materia y energía se realiza en una serie de etapas que constituyen lo que denominamos una cadena de alimentación.

El primer eslabón de toda cadena alimenticia es la captura de energía y de compuestos inorgánicos que las plantas verdes convertirán en materia orgánica mediante la fotosíntesis.

El tercer eslabón comprende todos los organismos heterótrofos que habitan esa área natural (carnívoros y herbívoros).

Como cuarto eslabón hay que considerar las bacterias y hongos, que desintegran la materia orgánica de los cadáveres.

El quinto y último eslabón lo forman las bacterias que transforman los compuestos inorgánicos en otros utilizables por las plantas verdes.

En cada eslabón de la cadena alimenticia se origina una pérdida de materia y energía. Como consecuencia de esta pérdida las cadenas alimenticias pueden representarse gráficamente como una pirámide de tres niveles llamada pirámide alimenticia.

A medida que se va ascendiendo en la pirámide los niveles son de menor tamaño

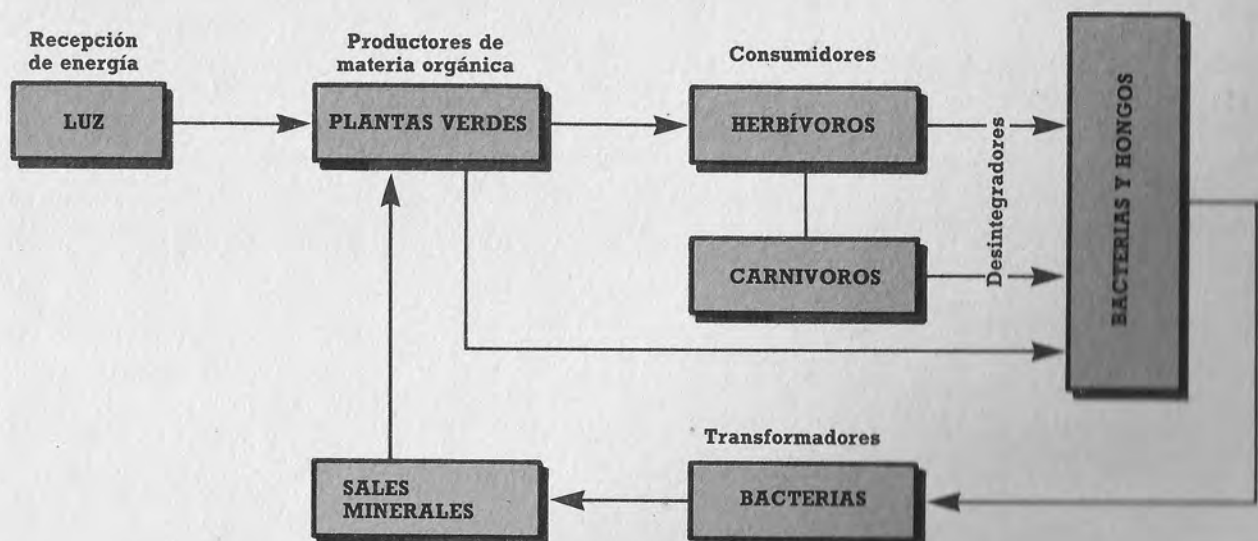


Fig. 1. Representación esquemática de una cadena alimenticia.

APRENDER CON EL ORDENADOR

(disminución de energía y materia) y es menor el número de individuos que los integran.

■ El programa

El programa está diseñado para ayudarte a entender la importancia que tienen estos conceptos biológicos y su mecanismo.

Representa gráficamente las relaciones en una pirámide alimenticia y realiza una serie de preguntas sobre los conceptos expuestos.

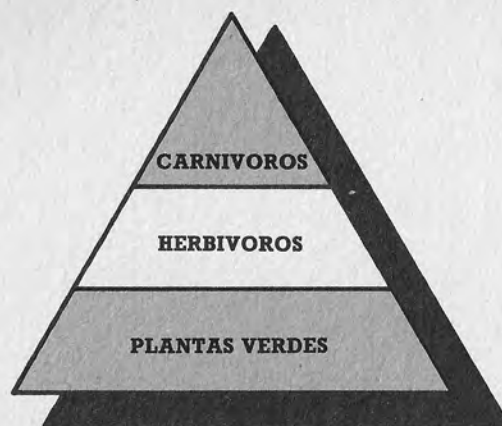


Fig. 2. Pirámide alimenticia.

```
10 REM *****
20 REM cadenas y piramides alimenticias
30 REM *****
40 CLS
50 PRINT
60 DIM p$(10,2)
70 DIM r$(10,3)
80 DIM s(10)
90 REM *****
100 REM * dibujo piramide *
110 REM *****
120 PRINT TAB(8); "PIRAMIDE ALIMENTICIA"
130 PLOT 0,100
140 DRAW 250,350
150 DRAW 500,100
160 DRAW 0,100
170 PLOT 50,150
180 DRAW 450,150
190 PLOT 100,200
200 DRAW 400,200
210 PLOT 150,250
220 DRAW 350,250
230 REM
240 REM niveles
250 REM
260 LOCATE 12,15:PRINT "Herbivoros"
270 LOCATE 12,18:PRINT "Plantas verdes"
280 LOCATE 12,12:PRINT "Carnivoros"
290 LOCATE 13,8:PRINT "Hombre"
300 FOR g=1 TO 3000:NEXT g
310 CLS
320 REM *****
330 REM * preguntas *
340 REM *****
350 FOR i=1 TO 10
360 FOR j=1 TO 2
370 READ p$(i,j)
380 NEXT j
390 NEXT i
400 FOR f=1 TO 10
410 FOR c=1 TO 3
420 READ r$(f,c)
430 NEXT c
440 NEXT f
450 FOR t=1 TO 10
460 READ s(t)
470 NEXT t
480 FOR n=1 TO 10
490 PRINT
500 PRINT p$(n,1)+p$(n,2)
510 PRINT
520 PRINT
530 FOR j=1 TO 3
540 PRINT TAB(4) j;". ";r$(n,j)
550 NEXT j
560 PRINT
570 PRINT
```

```

580 REM *****
590 REM * comprobacion respuesta *
600 REM *****
610 INPUT "respuesta";c
620 PRINT
630 PRINT
640 IF c=s(n) THEN PRINT "Correcto":GOTO 660
650 PRINT "Te has equivocado, es";s(n)
660 FOR m=1 TO 2000
670 NEXT M
680 CLS
690 NEXT n
700 REM *****
710 REM * preguntas *
720 REM *****
730 REM
740 DATA "Que posicion ocupan ","los ciervos en la piramide"
750 DATA "Las plantas ","verdes son"
760 DATA "Las cadenas ","alimenticias representan"
770 DATA "La produccion de materia ","organica la realizan"
780 DATA "El numero de eslabones"," de una cadena"
790 DATA "Una piramide alimenticia ","representa"
800 DATA "Un ejemplo de ","piramide es"
810 DATA "El proceso de obtencion ","de materia organica es"
820 DATA "Al ascender en la piramide ","el numero de individuos"
830 DATA "La desintegracion de materia ","organica la realizan"
840 DATA "Nivel superior"
850 DATA "Nivel intermedio"
860 DATA "Nivel inferior"
870 DATA "Autotrofas","Microorganismos"
880 DATA "Heterotrofas"
890 DATA "Intercambios energeticos"
900 DATA "Creacion de materia"
910 DATA "Transformaciones organicas"
920 DATA "Microorganismos","Bacterias"
930 DATA "Plantas verdes"
940 DATA "Depende de la piramide","Es variable"
950 DATA "Siempre es el mismo"
960 DATA "Una cadena alimenticia"
970 DATA "Un ecosistema"
980 DATA "Relacion entre animales"
990 DATA "Zorros, plantas, conejos"
1000 DATA "Conejos, zorros, plantas"
1010 DATA "Plantas, conejos, zorros"
1020 DATA "La biosintesis","La quimiosintesis","La fotosintesis"
1030 DATA "Disminuye","No varia","Aumenta"
1040 DATA "Bacterias","Hongos","Bacterias y hongos"
1050 DATA 2,1,1,3,2,1,3,3,1,3

```

Modificaciones para otros equipos

COMMODORE

40 print chr\$(147)
130 line 2,2,124,160
140 line 124,160,250,2
150 line 250,2,2,2
líneas 160, 170, 180, 190, 200, 210, 220 no se ponen

Para la modificación de las sentencias locate ver tomo I de esta colección

SPECTRUM

130 PLOT 2,2
140 DRAW 122,158
150 DRAW 126,-158
160 DRAW -148,0
LINEAS 170, 180, 190, 200, 210, 220 no se ponen
260 print at 8,13;"CARNIVOROS"
270 print at 12,12;"HERBIVOROS"
280 print at 15,12;"PLANTAS VERDES"

IBM

45 screen 2
130 line (0,100)-(250,350)
140 line -(500,100)
150 line -(0,100)
160 line (50,150)-(450,150)
170 line (100,200)-(400,200)
180 line (150,250)-(350,250)
líneas 190,200,210,220 no se ponen
260 locate 15,12
270 locate 18,12
280 locate 12,12
290 locate 8,13

MSX

45 screen 1
130 line (2,2)-(124,160)
140 line (124,160)-(250,2)
150 line (250,2)-(2,2)
170,180,190,200,210,220 no se ponen

APRENDER CON EL ORDENADOR

MATEMATICAS

Aproximación de PI

En una circunferencia de radio unidad el lado del cuadrado inscrito es $l_1 = (2)^{-1/2}$ y su perímetro es $p_1 = 4 (2)^{-1/2}$

El octógono inscrito tiene de lado

$$l_2 = \sqrt{(l_1/2)^2 + (1 - l_1/2)^2} = \sqrt{l_1^2/2 + 1 - l_1}$$

Fig. 3. Poligonos inscritos en circunferencias.

y su perímetro será:

$$P_2 = 8 \cdot l_1$$

Fig. 4.

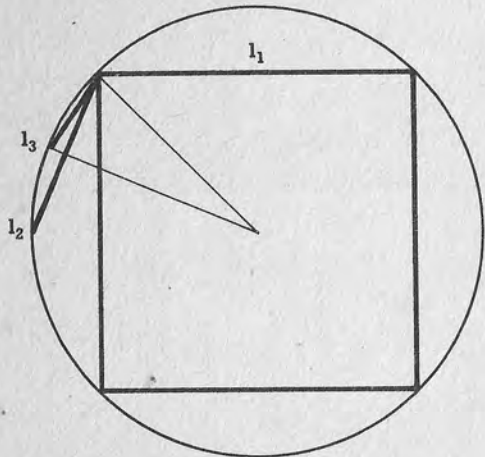


Fig. 5.

Si repetimos el proceso aumentando el número de lados del polígono inscrito en progresión geométrica de razón 2, la longitud del lado que resulta es

$$l_n = \sqrt{2 - \sqrt{4 - l_{n-1}^2}}$$

Fig. 6.

y el perímetro

$$P_n = 2^{n+1} \cdot l_n$$

Fig. 7.

Cuando n sea muy grande, el lado del polígono inscrito será muy pequeño y el polígono se confunde prácticamente con la circunferencia que lo contiene. Es decir, su perímetro tenderá a 2π . Ya que consideramos que $r=1$, en el límite obtendremos 2π .

Por ello, dividiendo el perímetro p_n entre 2 conseguiremos aproximaciones de π cada vez más exactas.

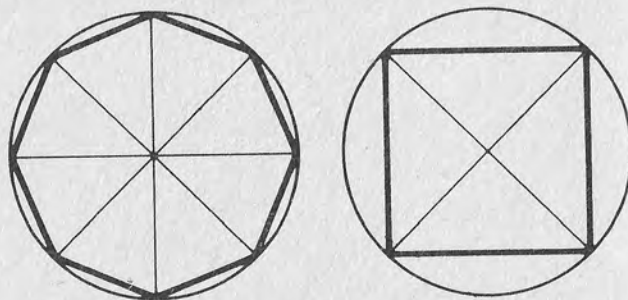


Fig. 8.

El programa

Consta de dos partes. En la primera dibuja el polígono inscrito e indica el valor que obtiene para π . Este proceso se repite varias veces, aumentando el número de lados del polígono. Como verás, el polígono es cada vez más parecido a una circunferencia.

En la segunda te presenta una tabla como la siguiente, con todos los resultados

N.º Lados	Aproximación de π
8	3.06146746
16	3.12144515
32	3.13654851
64	3.14033120
128	3.14127719
256	3.14151398
512	3.14157713
1024	3.14159656
2048	3.14159655

```

10 REM *****
20 REM * aproximacion de pi*
30 REM *****
40 CLS
50 DIM a(14,2)
54 REM
55 REM inicializar variables
56 REM
60 LET l=2*0.5
70 LET r=100
80 LET a=PI/6
84 REM *****
85 REM * representacion de poligonos *
86 REM *****
90 FOR i=2 TO 10
100 PRINT "N.lados", "Aproximacion de pi"
110 LET l=(2-(4-1^2)^0.5)^0.5
120 LET a(i,1)=2^(i+1)
130 LET m=1*a(i,1)
140 LET a(i,2)=m/2
150 PRINT a(i,1),a(i,2)
160 GOSUB 290
170 NEXT i
180 REM *****
190 REM * escritura de la tabla *
200 REM *****
210 PRINT "N.lados", "Aproximacion de pi"
220 PRINT "_____", "_____"
230 PRINT
240 FOR n=2 TO 10
250 PRINT a(n,1),a(n,2)
260 NEXT n
270 END
280 REM *****
290 REM * subrutina de dibujo *
300 REM *****
310 LET p=2*PI/a(i,1)
320 ORIGIN 320,200
330 MOVE r*COS(a),r*SIN(a)
340 FOR k=a+p TO a+2*PI+0.02 STEP p
350 LET x2=r*COS(k)
360 LET y2=r*SIN(k)
370 DRAW x2,y2
380 LET x1=x2
385 LET y1=y2
390 NEXT k
400 REM *****
410 REM * retardador *
420 REM *****
430 IF a(i,1)<100 THEN LET z=1700:GOTO 450
440 LET z=10
450 FOR e=1 TO z
460 NEXT e
470 CLS
480 RETURN

```

Modificaciones para otros equipos

COMMODORE

```

10 print chr$(147)
320 let x1=127+r*cos(a)
330 let y1=87+r*sin(a)
350 let x2=127+r*cos(a)
360 let y2=87+r*sin(a)
370 line x1,y1,x2,y2
460 print chr$(147)

```

SPECTRUM

```

320 let x1=127+r*cos(a)
330 let y1=87+r*sin(a)
335 plot x1,y1
350 let x2=127+r*cos(a)
360 let y2=87+r*sin(a)

```

IBM

```

45 screen 2
320 let x1=200+r*cos(a)
330 let y2=100+r*sin(a)
335 pset(x1,y1)
350 let x2=200+r*cos(a)
360 let y2=100+r*sin(a)
370 line (x2,y2)

```

MSX

```

45 screen 1
320 let x1=150+r*cos(a)
330 let y1=90+r*sin(a)
335 pset (x1,y1)
350 let x2=150+r*cos(a)
360 let y2=90+r*sin(a)
370 line -(x2,y2)

```

SOCIEDAD

Lección de inglés

El conocimiento del idioma inglés se hace más imprescindible día a día. Te proponemos hoy una forma fácil y divertida de empezar a estudiar este idioma.

El programa consta de dos partes: vocabulario y examen. En la primera el ordenador te presentará en la pantalla un vocabulario con las palabras en inglés y su correspondiente traducción al castellano.

Cuando consideres que ya has memo-

rizado correctamente todo el vocabulario, pulsa la letra "s".

¿Lo ves? Ahora el ordenador presentará una palabra en español para que tú escribas su traducción al inglés. Si la respuesta es correcta, te propondrá otra palabra. Si te has equivocado al contestar, marcará tu respuesta con un asterisco y te recordará cuál es la contestación correcta.

Al final te indicará el número de fallos que has cometido y te ofrecerá la oportunidad de repetir la lección.

No olvides responder siempre con minúsculas.

```

10 REM *****
20 REM * leccion de ingles *
30 REM *****
40 DIM p$(15,2)
50 DIM c(15)
60 FOR i=1 TO 15
70 FOR j=1 TO 2
80 READ p$(i,j)
90 NEXT j
100 NEXT i
110 LET m=0
120 CLS
130 FOR n=1 TO 15
140 LET c(n)=0
150 NEXT n
160 REM *****
170 REM * escribe la leccion *
180 REM *****
190 PRINT TAB(10)"Leccion de ingles"
200 PRINT TAB(10)" "
210 PRINT
220 PRINT TAB(5)"Breakfast";TAB(20)"Desayuno"
230 PRINT TAB(5)" ";TAB(20)" "
240 PRINT
250 FOR i=1 TO 15
260 PRINT TAB(5) p$(i,1);TAB(20) p$(i,2)
270 NEXT i
275 PRINT
280 PRINT "Cuando la hayas aprendido"
290 PRINT "pulsa la 's'"
300 INPUT w$
310 IF w$(">"s") THEN GOTO 300
320 CLS
324 REM *****
325 REM * preguntas *
326 REM *****
330 FOR k=1 TO 15
340 LET t=INT(RND*15)+1
350 IF c(t)=t THEN GOTO 340
360 LET c(t)=t
370 PRINT p$(t,2);TAB(10)" ";
380 INPUT r$
390 PRINT
400 IF r$=p$(t,1) THEN GOTO 430
410 LOCATE 22,k:PRINT "*" + p$(t,1)
414 REM *****
415 REM * contar los fallos *
416 REM *****
420 LET m=m+1
430 NEXT k
440 PRINT "Has tenido";m;"fallos"
450 PRINT
460 INPUT "Quieres intentarlo otra vez (si/no)";l$
470 IF l$="si" THEN GOTO 110
480 END
484 REM *****
485 REM * vocabulario *
486 REM *****
487 REM
490 DATA "milk","leche","bread","pan"

```

```

500 DATA "honey","miel","biscuit","galleta"
510 DATA "tea","te","coffee","cafe"
520 DATA "cup","taza","juice","zumò"
530 DATA "jam","mermelada","teapot","tetera"
540 DATA "sugar","azucar","saucer","platillo"
550 DATA "fruit","fruta","breakfast","desayuno"
560 DATA "knife","cuchillo"

```

Modificaciones para otros equipos

COMMODORE

```

120 print chr$(147)
320 print chr$(147)
340 let t=int(rnd(1)*15)+1

```

SPECTRUM

```

410 print at K,22;" "+P$(t,1)

```

IBM

```

410 locate k,22

```

MSX

```

340 let t=int(rnd(1)*15)+1

```

PARA LOS MAS JOVENES

■ Números omitidos

Este programa está pensado para que los más pequeños practiquen matemáticas.

El ordenador te irá proponiendo una serie de sumas (o restas) en las que se omite un operando, que aparecerá marcado con un signo de interrogación.

Ejemplo:

$$234 + ? = 563$$

$$? - 4 = 6$$

$$169 + 7 = ?$$

Deberás indicar el número que falta. Si te equivocas, el ordenador te enseñará la solución.

```

10 REM *****
20 REM * numeros omitidos *
30 REM *****
40 DIM r(3)
50 CLS
60 PRINT
70 PRINT
80 PRINT
90 LET r(1)=INT(RND*90)+1
100 LET r(2)=INT(RND*90)+1
110 LET s=INT(RND*2)
120 IF s=0 THEN LET s=-1
130 LET r(3)=r(1)+r(2)*s
140 IF r(3)<0 THEN GOTO 90
150 LET om=INT(RND*3)+1
160 REM *****
170 REM * escribir preguntas *
180 REM *****
190 IF om=1 THEN PRINT " ? ";GOTO 210
200 PRINT r(1);
210 IF s=-1 THEN PRINT "-";GOTO 230
220 PRINT "+";
230 IF om=2 THEN PRINT " ? =";GOTO 250
240 PRINT r(2);"=";
250 IF om=3 THEN PRINT " ? ";GOTO 270
260 PRINT r(3)
270 PRINT
280 PRINT
290 PRINT
300 INPUT "Que numero falta ";c
310 IF c<>r(om) THEN GOTO 370
315 REM
320 REM *****
330 REM * comprobar respuesta *
340 REM *****
345 REM
350 PRINT "Muy bien, la respuesta";
360 PRINT " es correcta";GOTO 400
370 PRINT "Te has equivocado, era";r(om)
380 REM *****
390 REM * retardador *
400 REM *****
410 FOR i=1 TO 2000
420 NEXT i
430 GOTO 50

```

Modificaciones para otros equipos

COMMODORE

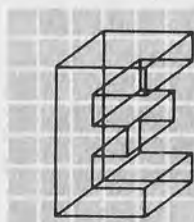
```
50 print chr$(147)
90 let r(1)=int(rnd(1)*90)+1
100 let r(2)=int(rnd(1)*90)+1
```

```
150 let om=int(rnd(1)*3)+1
```

MSX

```
90 let r(1)=int(rnd(1)*90)+1
100 let r(2)=int(rnd(1)*90)+1
150 let om=int(rnd(1)*3)+1
```

LA MAQUINA DE CALCULAR DE LEIBNIZ



N 1671, Leibniz acababa de crear su máquina de calcular. Parece increíble que estos ingenios mecánicos, típicos de relojero, causaran el mínimo interés en filósofos, matemáticos, científicos, en general. No es extraño.

Toda labor científica lleva implícitos cálculos triviales repetitivos, que no ofrecen interés alguno, pero cuya exactitud y fiabilidad son fundamentales para toda labor investigadora. Estos cerebros pensantes de la época se enfrentaban con los problemas, y los resolvían. Probablemente, si la madre de Leibniz hubiera sido coja o paralítica, su hijo hubiera inventado una silla de ruedas capaz de realizar los movimientos más impensables. Pero no fue así, y sólo tuvo que ocuparse de ella la Inquisición, que la acusó de haber procreado a su hijo con el diablo. ¿Cómo era posible que su hijo se interesara por semejantes asuntos?

La verdad, es que Leibniz estaba interesado en ese momento en Astronomía, ciencia que requiere multitud de cálculos. También era muy corriente la necesidad de calcular tablas de las funciones trigonométricas elementales.

Su máquina era una innovación de la máquina de Pascal, pero más sencilla de utilizar que aquella. La máquina de Pascal era esencialmente una sumadora, y Leibniz mejoró enormemente la multiplicación. (La máquina de Pascal necesitaba ir registrando el número a sumar en cada paso.) La primera máquina de Leibniz fue construida en 1671. El mismo describió su máquina en un informe escrito en latín, quince años después:

"La máquina está formada por dos par-

tes distintas, una para la suma (o resta) y otra para la multiplicación (o división). Estas dos partes se ensamblan. El mecanismo de sumar es idéntico al de Pascal, sin embargo, respecto de la multiplicación, se ha considerado que..."

Describamos a continuación la máquina de Leibniz, según sus propias palabras, escritas en el informe de que hablamos anteriormente.

La máquina tiene tres tipos de ruedas: las de sumar, las del multiplicando y las del multiplicador. Las ruedas del multiplicando son todas del mismo tamaño, e iguales también a las de la suma. Van provista de dientes, pero éstos son móviles, pueden sobresalir o hundirse, de forma que en unos casos la rueda tenga cuatro dientes, en otra, seis, etc. Por ejemplo, para un multiplicando 365, debemos utilizar tres ruedas, la de la derecha, con cinco dientes, la del medio, con seis, y la de la izquierda, con tres. Para que esto pueda realizarse, debe disponerse de un dispositivo "peculiar".

Este dispositivo "peculiar" constituye precisamente la parte revolucionaria de la máquina: la rueda escalonada (véase el dibujo). Esta rueda consiste esencialmente en un tambor cilíndrico que lleva nueve varillas cuya longitud es variable. Se utiliza también otra rueda menor, que engrana con el tambor anterior. Esta pequeña rueda puede desplazarse por su eje (paralelo al del cilindro anterior, de forma que al atacar, engrane con un número de dientes que puede variar según su posición. La rueda puede atacar en nueve posiciones diferentes.

Las ruedas del multiplicando constan, pues, de varias "ruedas" de Leibniz montadas sobre un eje común, y los dígitos del multipli-

Leibniz junto a una de sus máquinas de calcular.



cando se registran moviendo los pulsadores asociados con las ruedas. A su vez, cada cilindro escalonado está conectado con su correspondiente "rueda de sumar". Las multiplicaciones se realizan girando las ruedas escalonadas para cada dígito y moviendo las ruedas una posición a la izquierda en cada fase digital.

Leibniz tenía otro tercer juego de ruedas para un automatismo más avanzado aún de su máquina. Estas ruedas del "multiplicador" son nueve y tienen un diámetro proporcional al valor del dígito. Los juegos de ruedas de la parte multiplicador se conectan por cadenas articuladas haciéndose las conexiones cuando se introduce el multiplicador. De esta forma, el resultado de multiplicar el multiplicando completo por uno de los dígitos del multiplicador se envía al registro de resultados, ocurriendo lo mismo con los demás dígitos del multiplicando. Para finalizar la operación, se realiza la suma de los tres registros, obteniéndose la multiplicación prácticamente por el mismo sistema que se utiliza para hacerlo sobre el papel.

En el mismo informe, Leibniz alude a Pascal, para marcar diferencias entre ambas

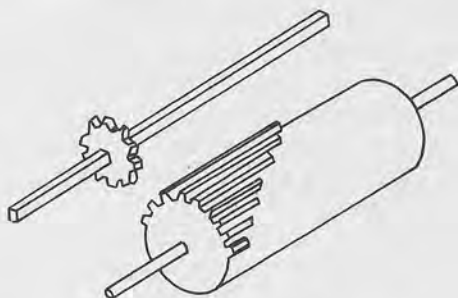
máquinas. Así dice que "la máquina de Pascal es fruto del ingenio más afortunado, pero sólo facilita la suma y resta, requiriendo ciertos cálculos para las multiplicaciones. Realmente no es una máquina práctica, sino más bien para recreo del curioso. Y también "nuestra máquina será de utilidad a todos los que empleen el cálculo, que, como es sabido, son los directores de empresas financieras, administradores de fincas, técnicos comerciales, peritos geógrafos, navegantes, astrónomos y todos aquellos cuya actividad esté relacionada con las Ciencias Matemáticas".

La máquina de Leibniz también ha sido copiada y mejorada hasta la saciedad. Muchísimas registradoras de entre las más conocidas han utilizado este tipo de tecnología. Leibniz construyó bastantes máquinas, aunque también el intento de mecanizar los cálculos volvió a ser un fracaso estrepitoso. Algunas de las máquinas de Leibniz se pueden ver en Hannover, donde pasó los últimos años de su vida.

Leibniz se interesó mucho por el sistema binario, que estudiaría más a fondo Boole. Sin embargo, su calculadora funcionaba en base diez. Si hubiera adoptado este sistema,

es probable que el ordenador (al menos los primeros intentos de ordenador) se hubieran llevado a cabo cien años antes.

Sin embargo, durante cincuenta o sesenta años, apenas si se llevaron a cabo modificaciones de la máquina. Los materiales de la



Rueda escalonada de Leibniz.

época no eran lo suficientemente fiables, y las máquinas no eran eficaces. Hasta 1810, no se llegó a un prototipo capaz de comercializarse con cierto éxito. Los tiempos habían cambiado, y las máquinas estaban de moda. Charles Xavier Thomas de Colmar (Alsacia) trabajaba en una empresa de seguros, que sigue existiendo en nuestros días, Le Soleil. La máquina le era muy útil en su trabajo, y Thomas pensó que muchos otros como él comprarían este tipo de máquina si fuera sencilla y fiable. Arriesgó esfuerzo y dinero en crear el prototipo, y, en este caso, finalmente, la operación fue un éxito completo, vendiéndose la fantástica cantidad de mil quinientas máquinas. El sistema adoptado era la rueda escalonada de Leibniz. En este tipo de máquina, los sumandos se registraban moviendo unas palancas dentro de unas ranuras, la cantidad seleccionada. Las ranuras llevan diez marcas correspondientes a los diez dígitos del sistema decimal. El movimiento de una de las palancas hace que la rueda de piñón se deslice por su eje, paralelo al cilindro escalonado de Leibniz. En este caso se han eliminado las "ruedas del multiplicador", siendo sustituidas por un registro que cuenta el número de vueltas de la manivela.

En 1875, Baldwin y WT Ohdner patentaron una variante de la "rueda de Leibniz". Tuviron mucho éxito, y sus máquinas se fabricaron en muchos países.

Cinco años antes de la máquina de Baldwin, un francés introdujo una técnica que, tras sufrir perfeccionamientos sucesivos, sería fundamental en el desarrollo de los ordenadores: la ficha, o cartulina perforada con ciertos códigos.

¿Sabía usted que...

No sabemos lo que podía costar la calculadora de Pascal, o la de Leibniz. Sin embargo, en los años treinta, una calculadora podía valer unos 1.200 dólares (precio astronómico, equivalente a dos coches). Ya en los años cincuenta, el precio había bajado a un octavo del precio anterior, pudiéndose comprar por la cuarta parte del valor de un coche. Calculadoras que en 1970 importaban unos 225 dólares bajó un tercio de su valor en sólo dos meses, y a la mitad, en los dos meses siguientes.

La primera calculadora electrónica data de 1963 y era británica, aunque fabricada bajo licencia americana y japonesa. Casi desde el primer momento, los japoneses se impusieron en la fabricación de calculadoras, probablemente porque su mano de obra era más barata, y el personal trabajador menos conflictivo, lo que abarata los costes sustancialmente.

Hagamos una pequeña biografía de Leibniz. No es de extrañar que la Inquisición se ocupara de su madre, pues es increíble cómo pudo adquirir tantos conocimientos tan diversos, y desarrollar otros tantos, tan distintos que parecen hasta contrapuestos.

Leibniz nació en Leipzig, en 1646, y murió en Hannover, en 1716. Fue todo un erudito en lenguas clásicas, Filosofía y Literatura antigua (se interesó especialmente por el hebreo), Teología escolástica, Matemáticas, doctrinas de Descartes, Gazzendi, Hobbes, Locke, Kepler y Galileo. ¿Quizá le faltó más interés por el arte, como Pintura y Música para ser completo? ¿O acaso no aprovechaba bien su tiempo, y no pudo aprender más? Viajó y se relacionó con los sabios de su época. Fundó la Academia de Ciencias de Berlín en 1700 (naturalmente, fue su primer presidente). Formuló el cálculo infinitesimal, en 1676 (independientemente de Newton). Pero no dedicó toda su vida a la razón, también se interesó por la Religión, aunque, debemos decirlo, también a través de la razón. Intentó la unión de todas las religiones cristianas. Fue un verdadero talento universal. Cultivó todas las ciencias, y al mismo tiempo se esforzó por conseguir una síntesis armoniosa que se revela en su sistema filosófico: concepción espiritualista y dinámica de todo el ser, a base de elementos primordiales llamados mónadas, organizados en un orden preestablecido por una mónada suprema: Dios.

PEQUEÑA HISTORIA DE LA INFORMATICA

En su juventud, Leibniz quedó deslumbrado por las lecturas de Lull. A los diecinueve años leyó su *Dissertio de arte combinatoria*, en la que cree descubrir el germen de un álgebra universal. No es extraño que se sintiera atraído por ese extraño personaje medieval, interesante y extravagante. Sin embargo, ya en los últimos años de su vida y con más experiencia (al fin y al cabo, hasta Leibniz de joven era inexperto), Leibniz criticó muchas de las ideas del *Dissertio de arte combinatoria*. En sus escritos, contradiciendo a Lull, Leibniz sí consideró que estaba acertado en muchos aspectos, aunque, desde luego, lo consideró menos profundo y serio que Descartes.

Tampoco Leibniz compartía el empirismo de Locke, ni la misma concepción dinámica de Descartes. Aunque Leibniz vivió muchos años, y escribió bastante, su producción está

muy dispersa, y se han perdido muchos de sus trabajos.

Además de Leibniz, otros grandes sabios han estado siempre interesados en saber si en el futuro se podrán inventar máquinas que ejecuten las tareas más creadoras, típicas del hombre. Entre ellas están la de experimentar, sugerir hipótesis, simular situaciones, etc. Esta aspiración la han tenido desde hace varios siglos muchos filósofos, desde Lull, Leibniz, Bacon, etc. En el momento actual, casi se ha conseguido el objetivo. La Inteligencia Artificial intenta resolver ese dilema de la creatividad.

Existe, sin embargo, una idea de Leibniz que no compartimos en absoluto, y la defenderemos a capa y espada: para Leibniz, este mundo es el mejor de los posibles. Para nosotros, desde luego, no.

EL SOFTWARE DE LOS ACELERADORES DE PARTICULAS



OMO hemos dicho anteriormente, los programas utilizados en el control y manejo de los aceleradores de partículas y todo el material anejo, están escritos casi exclusivamente en FORTRAN. Existen dos estructuras diferentes, la matriz de elementos homogéneos y el bloque "common" (de entidades no homogéneas). Este último bloque "common" no puede manipularse como entidad simple. Como puede esperarse, la complejidad de los datos obtenidos de los detectores, y las estructuras relacionadas con ellos, requieren un sistema de tratamiento de éstos, más complejo que lo que el FORTRAN puede ofrecer. Existen ciertas relaciones entre los elementos que el FORTRAN no puede expresar, ya que no dispone de un tipo de datos para utilizarlo como indicador. Eso, sin considerar la falta de ahorro de espacio. Si consideramos, por ejemplo, un acontecimiento físico cualquiera, observaremos que consiste en uno o más vértices, del que emergen un cierto número de pistas, cada una con un número de señales variable (y estas señales, a su vez, provienen de un número de detectores también variable). Las asociaciones lógicas entre estas entidades se pueden expresar mejor mediante indicadores, que conecten cada uno un banco de datos. De este modo, se habrá creado una estructura dinámica, etapa por etapa, comenzando por las señales que se obtienen de los detectores, pasando a los segmentos de pista para terminar alcanzando los vértices. Aquellas asociaciones que son alternativas, se pueden representar por varios conjuntos de indicadores alternativos entre bancos de datos.

Naturalmente, existen varios paquetes es-

pecializados, que adecúan la estructura del Fortran específicamente para la física nuclear. En el CERN, el centro europeo situado cerca de Ginebra, se ha desarrollado el lenguaje ZEBRA, basado en la versión 77 del Fortran. A su vez, en DESY (EE.UU.), se ha desarrollado el BOS77, también basado en esa versión del Fortran. Tanto uno como otro paquete utilizan pilas de almacenamiento para solicitar bancos individuales. También se pueden relacionar varios bancos. La estructura puede leerse y escribirse en un medio externo. En el paquete Zebra, las estructuras se agrupan a su vez en unidades más amplias, pudiéndose hacer en ella "garbage collection" (identificación y eliminación de información innecesaria). Naturalmente, todos estos paquetes van provistos de programas de ayuda para usuarios, y otros de depuración, para ayudar al usuario a manejar la enorme complejidad de los datos.

Como estos datos deben ser manejados desde un único programa, se deben establecer mecanismos para que los datos se puedan intercambiar, en particular entre programas que funcionen en ordenadores diferentes con longitudes de palabra diferentes, y, en suma, representaciones incompatibles. Para resolver este problema de incompatibilidad han aparecido, a su vez, otros paquetes. Uno de ellos es Epio. Este paquete está formado por un conjunto de rutinas que permiten convertir los datos a la representación interna IBM. A continuación, se escriben en un medio externo, en formato compatible, y por tanto, totalmente manipulable por el usuario. Gracias a ellos, por los registros de datos se pueden volver a leer en un ordenador diferente, y pueden, por tanto, ser reutilizados a conveniencia del usuario. Este tipo de programas se utilizan muchísimo en los laboratorios a

TEMAS MONOGRAFICOS DE VANGUARDIA

todos los niveles y para la gama amplia de ordenadores utilizados.

La lectura y escritura de datos, no estándar IBM, es muy complicada para sistemas IBM, y existen paquetes que intentan obviar estos problemas.

Como en Física de las partículas, se manejan enormes cantidades de datos, parámetros, etc., es importantísimo el manejo adecuado de bases de datos (de hecho, en la informática del futuro, los ordenadores estarán conectados a enormes bases de datos, que manejarán sin problemas, como ayuda en la toma de decisiones, simulaciones, etc.).

El almacenamiento de los datos es también otro problema muy importante. No hay que olvidar que el problema principal es el enorme aluvión de datos a procesar.

Los programas de control del acelerador son muy largos y complejos (cada uno puede tener en media 1.000 subprogramas). Además, como antes dijimos, deben ser utilizados en cierto número de ordenadores, posiblemente en países diferentes. Además, deben ser asequibles al personal que los utiliza.

En los primeros tiempos, los programas especializados para este tipo de problemas se realizaban en el propio laboratorio, por algún físico del equipo, interesado en la informática. Los programas iban creciendo, al ir creciendo las necesidades del laboratorio. Hoy en día, estas técnicas son impensables. Se ha hecho necesaria una especialización, y existen físicos dedicados exclusivamente a tareas informáticas. Se han creado cursos especializados donde estos físicos se forman, especialmente en análisis y diseño estructurados.

El lenguaje que se sigue utilizando es el Fortran, que se ha considerado el más adecuado a lo largo del tiempo. Naturalmente, se han ido mejorando las versiones utilizadas, pero en la selección de este lenguaje también se ha tenido en cuenta que, como ya hemos indicado, existe un intercambio de información enorme, entre los distintos laboratorios, a nivel mundial.

El mantenimiento de los programas es un problema importante. Los códigos fuente se modifican continuamente por distintos usuarios para satisfacer sus necesidades concretas; sin embargo, todas estas modificaciones no deben afectar a los demás usuarios, y por tanto, debe ser posible obtener versiones más antiguas que no contengan las modificaciones efectuadas. Hoy en día, existen paquetes co-

mercializados que solucionan estos problemas (aunque hace unos años, en los laboratorios de Física se utilizaban casi exclusivamente, los nacidos allí mismo). Con el tiempo, se van adquiriendo más y más paquetes estándar, que se van adaptando a la problemática concreta del control del acelerador.

Los gráficos siempre han ayudado a ver con más claridad los problemas y sus posibles soluciones. Por muy complicada que sea la fisión nuclear, tampoco es una excepción. Por tanto, se ha de considerar que para abordar cualquier problema, éste puede subdividirse y estudiarse sus partes por separado, terminando por volver a considerar el problema globalmente, es decir, en su totalidad. Los laboratorios de física de las partículas son pocos y muy costosos. Su mantenimiento también es muy caro, y a él atienden los altísimos presupuestos de que se dispone. Unos terminales gráficos no suponen apenas coste dentro de este enorme gasto anual.

Hace unos veinte años que comenzaron a adquirirse terminales gráficos en los laboratorios. La verdad es que, en un principio, el personal del laboratorio apenas si les veía utilidad. Sin embargo, con el tiempo se ha comprobado que son utilísimos y pueden aclarar muchas dudas, mostrando con claridad muchas relaciones y tendencias que apenas si se pueden deducir utilizando únicamente datos, valores y relaciones, aunque es verdad que los terminales gráficos son, por lo general, caros y, cuanto más atractivos, más.

En cuanto al software gráfico, en los primeros tiempos ocurría como con el normal, se creaba en el laboratorio. Del mismo modo, hoy en día existen innumerables y magníficos paquetes gráficos estándar, que utilizan por lo general interfaces Fortran entre los distintos dispositivos de software y hardware. Entre ellos, uno muy utilizado es el ISO estándar. Este paquete define varias primitivas, líneas, vectores, etc., y, además, con muchos atributos (estilo, color, intensidad, ancho, etc.). La salida no se hace a un dispositivo físico, sino a un metaarchivo, y éste proporciona un mecanismo, estandarizado sólo parcialmente para el almacenamiento y transferencia de estos ficheros gráficos a los distintos ordenadores. En cuanto a las entradas al programa, se pueden realizar, por ejemplo, mediante un joystick o un ratón.

El sistema más moderno consiste en considerar la información gráfica, como dividida en "pedazos" segmentos, de forma que éstos se



puedan mover por la pantalla sin que ello afecte a otros segmentos que se encuentren en ella. Para las aplicaciones gráficas orientadas a la física de las partículas, también se ha escogido el Fortran. En el CERN se utiliza el GKS, aunque tiene el grave inconveniente de que está limitado a dos dimensiones (lo que hoy en día resulta muy pobre, existiendo como existen, tantos magníficos programas gráficos que realizan representaciones en tres dimensiones). El laboratorio del CERN está mejorando

el GKS, para hacerlo tridimensional. El nuevo paquete se llama Phigs. Todavía el paquete no está disponible, y por tanto, se está utilizando otro creado totalmente en el laboratorio, el PIONS. Este paquete no sólo es tridimensional, sino que permite al usuario visualizar de forma interactiva las distintas partes de la estructura de los datos, almacenado por un sistema jerárquico, sin que para ello sea necesario acceder al programa de aplicación, que generó la estructura de los datos.

PARTICULAS ELEMENTALES

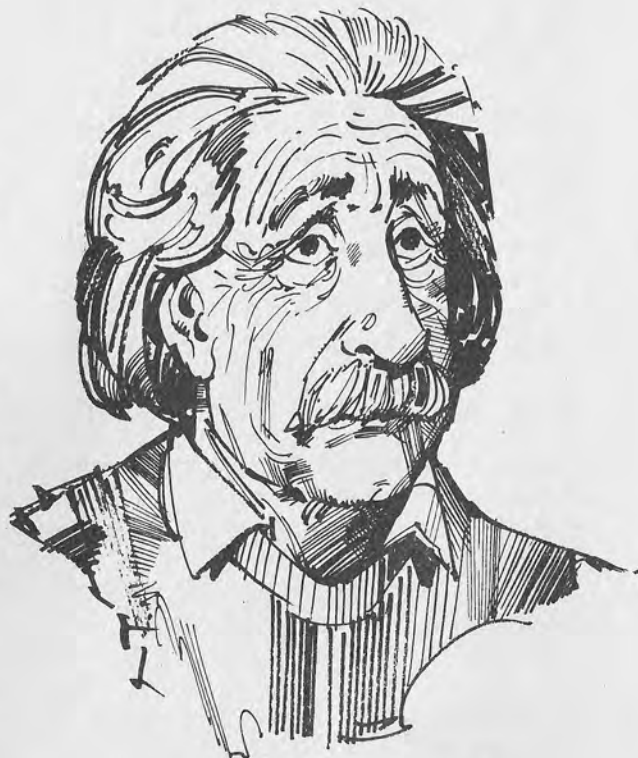
Partículas elementales. Partes últimas de la materia que se encuentran en los núcleos de los átomos o se forman por intercambio de materia-energía. Realmente, debería definirse cada partícula con respecto al proceso de división de que se trate. El término que debería utilizarse para designar a este tipo de partículas es "fundamental", no "elemental", ya que no se puede considerar una partícula elemental, si se desintegra en otras partículas. Todas las partículas elementales pueden definirse por ciertas características: su masa, su carga y su momento angular o spin, fundamentalmente. Entre las partículas elementales, podemos citar las siguientes:

Electrón. Corpúsculo de masa uno, carga $-e$ y spin $1/2$. Con su descubrimiento, se explicaron numerosos fenómenos de la Física clásica. La corriente eléctrica está formada por electrones y éstos pueden obtenerse libres poniendo el conductor al rojo (efecto Edison). La energía de un electrón es de varios eV. Sin embargo, pueden acelerarse hasta alcanzar energías enormes, en un betatrón. La radiactividad negativa, beta está compuesta por electrones de energías de varios Mev.

Fotón. Uno de los corpúsculos asociados a las ondas de la luz. Los fotones fueron descubiertos al intentar Einstein explicar el efecto fotoeléctrico. La dualidad corpuscular ondulatoria de las emisiones electromagnéticas fue explicada por De Broglie. Los fotones son neutros, no tienen masa (cero) y tienen un spin de unidad. Son corpúsculos asociados a vibraciones de ciertas frecuencias.

Estos fotones son absorbidos por los átomos en forma de luz o de rayos X, y en forma de rayos gama por los núcleos (siempre que exista un cambio de energía).

Protón. Una de las partículas integrantes del núcleo atómico, junto con los neutrones. Su masa es $1,672 \times 10^{-27}$ elevado a menos veinticuatro, y su carga está equilibrada por la de los electrones. Su spin es de $1/2$. En suma, es un ion de hidrógeno.



Einstein, padre de la Teoría de la relatividad.



Paul Langevin relacionó la Teoría de la relatividad desarrollada por Einstein con el problema de modificar átomos de hidrógeno para pasarlos a formas atómicas más complejas. Estudió los cambios de energía durante los procesos.

Antiprotón. Partícula cuya masa es algo mayor que la del protón, su carga es $-e$ y su spin $3/2$. Es el equivalente al positrón, en la teoría del electrón. Sobre esta partícula existen ciertas reservas. Fue obtenida en el

betatrón de Berkeley, en 1958. Su descubrimiento ayudó a establecer la existencia de una antipartícula por cada partícula, con masa y spines idénticos. Si la partícula no es estable, tampoco lo será la antipartícula. Las antipartículas se designan del mismo modo que las partículas, pero con una pequeña barra en su parte superior.

Neutrón. Fue descubierto dentro de la estructura del núcleo por el Dr. Chadwick. El neutrón tiene una masa de 1.838,7; no tiene carga y su spin es también $1/2$.

Antineutrón. Antipartícula de neutrón. Fue descubierta en 1956 en Berkeley. Estas partículas se forman por intercambio de cargas entre neutrones y antiprotones.

Neutrino. Partícula de masa 0, carga 0, y spin $1/2$, que permite explicar ciertas particularidades del espectro continuo de las energías del positrón.

Antineutrino. Antipartícula del neutrino, que permite explicar la desintegración de los neutrinos.

Mesones. Partículas encontradas en la radiación cósmica. Hoy en día se pueden producir en los aceleradores. Su masa es intermedia entre la del electrón y los nucleones. Su vida media es cortísima (de 10^{-8} a 10^{-10} segundos). Existen muchos tipos de mesones, μ , π , K, etc.

Hiperones. Partículas raras, de masa algo superior a la del neutrón (2.181, sin carga y con spin 1). El bombardeo de núcleos ligeros con partículas K produce hiperones.

VOCABULARIO DE INFORMATICA

Gap. Término inglés que significa espacio, separación. Suele tratarse de un separador de la información grabada en discos duros y flexibles, o en cintas magnéticas.

Garbage collection. Término inglés que indica la recuperación de partes de la memoria no empleadas por un programa, para su posterior utilización. Cuando la estructura está organizada en forma secuencial con indicadores, existen elementos no utilizados, o retales. El término designa, pues, la recopilación de estos elementos en una lista.

Gate. (Véase **Puerta**)

General Purpose Computer. (Véase **Ordenador de uso general**)

Generar. Producir un programa seleccionando subconjuntos de un conjunto de codificaciones de estructura, y bajo el control de ciertos parámetros.

Generador. Rutina de control que realiza una generación, por ejemplo, un generador de informes, un generador de Entrada/Salida.

Gestión de tareas. Conjunto de procedimientos y normas que regulan los trabajos que debe realizar el ordenador. Parte del hardware o software del ordenador que se encarga de la gestión de un determinado recurso. Puede tratarse de un gestor de memoria, de los canales de entrada o salida, etcétera.

Giga. Prefijo multiplicador. Equivale a mil millones (diez elevado a nueve).

Global. Término que se aplica normalmente a una variable que pertenece simultáneamente a varios bloques de un programa o a un programa principal y a sus subrutinas. En contraposición a variables locales. (Véase variable local y variable global).

Global variables. (Véase **variable global**).

Grabación magnética. Tecnología utilizada entre otras muchas cosas para almacenar la información. Dispone de unos "soportes" en la superficie, en los cuales se graban los datos mediante determinados campos magnéticos. Un ejemplo de estos dispositivos pueden ser las cintas y los discos flexibles. Hasta hace pocos años se han venido utilizando un tipo de memorias que utilizaban técnicas

magnéticas. Estos sistemas de almacenamiento de datos tienen la ventaja enorme de que se conservan perfectamente durante años.

Grabar. Transferir o almacenar información en un soporte cualquiera. Es sinónimo de salvar, guardar, escribir, etc.

Gráficos. Dibujos creados con el ordenador, tanto en la impresora o en un plotter, como en un monitor. Existen fundamentalmente varios tipos de gráficos: de alta resolución o gráficos completos, de media resolución y de baja resolución. Los gráficos de alta resolución son muy perfectos, pero su creación es lenta y costosa.

Gráfico, carácter. Carácter representado por un símbolo gráfico, en contraposición con carácter de control.

Gráfico símbolo. Símbolo producido para la creación de dibujos gráficos.

Grafo. Conjunto de elementos relacionados. Método gráfico que utiliza líneas para representar los valores del flujo.

Ground. (Véase **Tierra**).

Grupo, marca de. Marca que identifica el comienzo o el final de un conjunto de datos. Puede incluir palabras, bloques u otros elementos.

Grupo, carácter separador de. Carácter que actúa como separador para identificar la frontera entre dos elementos de un grupo.

Half adder. (ver **Semisumador**).

Half byte. (ver **Semibyte**).

Half duplex. En comunicaciones, sistema que sólo puede utilizar una dirección del canal. Es independiente de la transmisión. En contraposición a duplex.

Handler. (ver **Supervisor, manipulador**).

Handshake. (ver **Sincronización**).

Hard copy. Copia en papel de los resultados de un proceso o de lo que aparece en pantalla.

Hard disk. (véase **Disco duro**).

Hardware. Equipo físico, dispositivo mecánico, magnético, eléctrico o electrónico, en contraposición a los programas o software.

